

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2006-107514

(43)Date of publication of application : 20.04.2006

(51)Int.Cl.	G06F 15/80	(2006.01)
	G06F 9/52	(2006.01)
	G06F 9/54	(2006.01)

(21)Application number : 2005-292708

(71)Applicant : SONY COMPUTER
ENTERTAINMENT INC

(22)Date of filing : 05.10.2005

(72)Inventor : SUZUOKI MASAKAZU
YAMAZAKI TAKESHI

(30)Priority

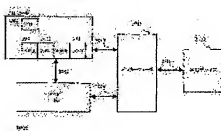
Priority number : 2004 959635 Priority date : 05.10.2004 Priority country : US

(54) SYSTEM AND DEVICE WHICH HAVE INTERFACE DEVICE WHICH CAN PERFORM DATA COMMUNICATION WITH EXTERNAL DEVICE

(57)Abstract:

PROBLEM TO BE SOLVED: To provide an architecture which enables various members on a network to share data and applications among them without an additional load in computing.

SOLUTION: A system is provided, which has a processing element (PE), an input/output (I/O) interface device, and a shared memory. The PE has at least one processing unit (PU) and also has one or more additional processing devices (APU). At least one of the APUs executes an I/O function by reading from or writing to an external device connected to the I/O interface device 2915. Data is exchanged between the APU and the I/O interface device 2915 by using a data level synchronization mechanism through the shared memory.



(19) 日本国特許庁(JP)

(12) 公開特許公報(A)

(11) 特許出願公開番号

特開2006-107514

(P2006-107514A)

(43) 公開日 平成18年4月20日(2006.4.20)

(51) Int. Cl.

F I

テーマコード (参考)

G06F 15/00 (2006.01)

G06F 15/00

G06F 9/52 (2006.01)

G06F 9/46 475C

G06F 9/54 (2006.01)

G06F 9/46 480Z

審査請求 有 請求項の数 27 O L (全 50 頁)

(21) 出願番号 特願2005-292708 (P2005-292708)
 (22) 出願日 平成17年10月5日(2005.10.5)
 (31) 優先権主張番号 10/859,635
 (32) 優先日 平成16年10月5日(2004.10.5)
 (33) 優先権主張国 米国(US)

(特許庁注: 以下のものは登録商標)

1. JAVA
2. イーサネット

(71) 出願人 395015319
 株式会社ソニー・コンピュータエンタテインメント
 東京都港区南青山二丁目6番21号
 (74) 代理人 100099324
 弁理士 鈴木 正剛
 (74) 代理人 100108604
 弁理士 村松 義人
 (74) 代理人 100111615
 弁理士 佐野 良太
 (72) 発明者 鈴木 雅一
 東京都港区南青山二丁目6番21号 株式会社ソニー・コンピュータエンタテインメント内

最終頁に続く

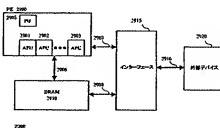
(54) 【発明の名称】 外部デバイスとデータ通信可能なインターフェイスデバイスを有するシステム及び装置

(57) 【要約】

【課題】 計算上の負担が附加されことなく、ネットワークの様々なメンバー間でのデータとアプリケーションの共用可能なアーキテクチャを提供する。

【解決手段】 処理エレメント (PE)、入出力 (I/O) インターフェイスデバイス及び共有メモリを有するシステムが提供される。PEは、処理ユニット (PU) を少なくとも一つ備え、かつ、一つ以上の附加処理装置 (APU) を備える。APUの少なくとも一つは、I/O インターフェイスデバイス 2915 に接続された外部デバイスに対して読み出しあるいは書き込みを行うことで I/O 機能を実行する。データは、APU と I/O インターフェイスデバイス 2915 との間で、データレベル同期メカニズムを用いて共有メモリを通じて交換される。

【選択図】 図 29



【特許請求の範囲】

【請求項 1】

外部デバイスへのデータ通信と外部デバイスからのデータ通信とを行うように動作可能なインターフェースデバイスと、

各々が前記データの格納をするように動作可能な、複数のメモリ・ロケーションを持つメモリとを含み、

前記インターフェースデバイスと前記メモリのうちの少なくとも1つが、前記メモリ・ロケーションの対応するメモリ・ロケーションと関連付けられた状態情報を格納するように動作可能であって、前記状態情報には、第1フィールドとアドレス・フィールドとが含まれ、この第1フィールドとアドレス・フィールドとは、与えられたメモリ・ロケーションに対して、前記関連する状態情報の前記第1フィールドの値が第1の値と等しくかつ前記関連する状態情報の前記アドレス・フィールドの値が第2の値と等しい場合に、前記メモリ・ロケーションへの書き込みオペレーションによって、前記メモリ・ロケーションに現在格納されているデータが、前記第2の値によって示されるアドレスへの書き込みが行われる、システム。

【請求項 2】

前記アドレスは前記外部デバイスと関連付けられている、請求項1記載のシステム。

【請求項 3】

前記アドレスは、前記インターフェースと通信を行うプロセッサと関連付けられているメモリ・ロケーションと関連付けられている、請求項1記載のシステム。

【請求項 4】

データアクセス用に前記メモリに結合され、かつ前記インターフェースと結合されているプロセッサを含み、前記プロセッサは前記外部デバイスと関連付けられた要求を発行するように動作可能である、請求項1又は2又は3記載のシステム。

【請求項 5】

少なくとも1つの前記要求は、前記外部デバイスからのデータの読み出しに対するものである、請求項4記載のシステム。

【請求項 6】

少なくとも1つの前記要求は、前記外部デバイスへのデータの書き込みに対するものである、請求項4記載のシステム。

【請求項 7】

前記メモリは、前記プロセッサと前記インターフェースデバイスとの間に前記要求を送るように動作可能である、請求項4又は5又は6記載のシステム。

【請求項 8】

前記外部デバイスは、前記インターフェースデバイスと結合されている、請求項1～7のいずれかに記載のシステム。

【請求項 9】

前記インターフェースデバイスは、DMA伝送を用いて、前記外部デバイスと前記メモリとの間でのデータ通信を行う、請求項1～8のいずれかに記載のシステム。

【請求項 10】

複数のメモリ・ロケーションを持つメモリと、

少なくとも1つの第1のタイプと第2のタイプの要求を発行するように動作可能で前記メモリと結合されたプロセッサと、

インターフェースデバイスと、を含み、前記インターフェースデバイスは、前記第1のタイプの要求を受信したことに応答して、前記外部デバイスから前記メモリヘデータを伝送し、前記メモリへ前記データを格納するように動作可能であり、かつ、前記インターフェースデバイスは前記第2のタイプの要求を受信したことに応答して、前記メモリのストレージから前記外部デバイスヘデータを伝送するように動作可能であって、

前記メモリの少なくとも1つのメモリ・ロケーションと、前記インターフェースデバイスとは、ブロッキング状態を含む、複数の異なるメモリの状態をサポートし、このサポー

トでは、前記ブロッキング状態にある与えられたメモリ・ロケーションへの書き込みオペレーションによって、現在、その中に格納されているデータが、前記与えられたメモリ・ロケーションに関連付けられたアドレスに書き込まれる、システム。

【請求項 11】

前記アドレス値は前記外部デバイスを表す、請求項 10 記載のシステム。

【請求項 12】

前記プロセッサと関連付けられたローカル・メモリを更に含み、前記アドレス値は前記ローカル・メモリのメモリ・ロケーションを表わす、請求項 10 又は 11 記載のシステム。

【請求項 13】

前記第 1 のタイプの前記要求は、前記外部デバイスからのデータの読み出しに対するものである、請求項 10 又は 11 又は 12 記載のシステム。

【請求項 14】

前記第 2 のタイプの前記要求は、前記外部デバイスからのデータの書き込みに対するものである、請求項 10 又は 11 又は 12 記載のシステム。

【請求項 15】

前記外部デバイスは前記インターフェースデバイスと結合されている、請求項 10 ～ 14 のいずれかに記載のシステム。

【請求項 16】

前記インターフェースデバイスは、DMA 伝送を用いて、前記外部デバイスと前記メモリとの間でデータ通信を行う、請求項 10 ～ 15 のいずれかに記載のシステム。

【請求項 17】

前記プロセッサは、プロセッシングユニット (PU) と、前記要求を発行するように動作可能な、少なくとも 1 つの付加処理ユニット (APU) とを含む、請求項 10 ～ 16 のいずれかに記載のシステム。

【請求項 18】

前記インターフェースデバイスは、前記メモリ・ロケーションへのアクセスを制御するための保護テーブルを含む、請求項 17 記載のシステム。

【請求項 19】

前記 PU は、前記保護テーブルの値を初期化する、請求項 18 記載のシステム。

【請求項 20】

外部装置とプロセッサ間を相互接続する装置であって、保護テーブルを含み、前記保護テーブルは前記プロセッサによってアクセス可能な共用メモリの領域を示す情報を格納するように動作可能であって、伝送コントローラを含み、前記伝送コントローラはプロセッサからの要求にตอบสนองして、前記外部デバイスと前記共用メモリの前記アクセス可能な領域のうちの少なくとも 1 つとの間のデータ伝送を制御するものであって、かつ、伝送バスを含み、前記伝送バスは前記伝送コントローラによる制御の下で、前記外部デバイスと、前記共用メモリの前記少なくとも 1 つのアクセス可能な領域との間に前記データを伝送するように動作可能であり、

前記伝送コントローラは、メモリ・ロケーションの最新のメモリ状態によって、前記少なくとも 1 つのアクセス可能な領域のメモリ・ロケーションと外部デバイスとの間のデータ伝送を同期して制御するよう動作可能であり、この制御では、最新のメモリ状態がブロッキング状態である場合、メモリ・ロケーションへの書き込みオペレーションにより、現在メモリ・ロケーションに格納されているデータが、前記メモリ・ロケーションに関連付けられたアドレスへ書き込まれる、装置。

【請求項 21】

インターフェースデバイスを介して、プロセッサと外部デバイスとの間にデータ伝送を行う方法であって、

前記外部デバイスと前記プロセッサに対してローカルであるメモリとの間で、前記デー

タを、複数のメモリ・ロケーションを持つ共用メモリを介して前記プロセッサへ伝送し、前記共用メモリは、ブロッキング状態を含む複数の異なるメモリ状態をサポートし、前記ブロッキング状態にある与えられたメモリ・ロケーションへの書き込みオペレーションにより、現在前記メモリ・ロケーションに格納されているデータが、前記所望のメモリ・ロケーションと関連付けられたアドレス値へ書き込まれる、方法。

【請求項 2 2】

前記プロセッサは第 1 プロセッサであり、更に、

第 2 プロセッサからインターフェースデバイスへ、インターフェースデバイスの保護テーブルにおいて用いるための値を送り、前記値は前記共用メモリの一部へのアクセスを制御する、請求項 2 1 記載の方法。

【請求項 2 3】

前記外部デバイスは、

外部デバイスを通じて、伝送されるデータを要求するコマンドを受信し、

前記コマンドに関連付けられたアドレス値が有効であるかどうかを判断するために、保護テーブルの値をチェックし、

アドレス値が有効である場合は、前記初期化ステップを行う、請求項 2 1 又は 2 2 記載の方法。

【請求項 2 4】

前記関連するアドレス値は前記外部デバイスを示す、請求項 2 1 又は 2 2 又は 2 3 記載の方法。

【請求項 2 5】

前記関連するアドレス値は前記プロセッサと関連付けられたローカル・メモリを示す、請求項 2 1 又は 2 2 又は 2 3 記載の方法。

【請求項 2 6】

デバイスであって、

要求や応答を複数の要求／応答チャネルを介して転送するための第 1 パスを含み、各要求／応答チャネルはプロセッサと関連付けられており、各要求／応答チャネルは、前記関連付けられたプロセッサと外部デバイスとの間にデータ通信の要求を転送するものであり、

メモリとのデータ通信を行う第 2 パスを含み、前記第 2 パスはデータレベル同期を用いて前記データの通信時に用いられるものであり、前記メモリは複数のメモリ・ロケーションを含み、前記各メモリ・ロケーションは、ブロッキング状態を含む複数の異なるメモリ状態をサポートするものであって、前記ブロッキング状態にある与えられたメモリ・ロケーションへのデータの書き込みオペレーションにより、現在前記与えられたメモリ・ロケーションに格納されているデータが、前記与えられたメモリ・ロケーションと関連付けられているアドレス値へ書き込まれる、デバイス。

【請求項 2 7】

前記第 1 パスは、前記プロセッサによりアクセス可能なメモリの一部を示す前記値を、保護テーブルで用いられるように転送する、請求項 2 6 記載のデバイス。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、同時係属中であり、本発明の譲受人に譲渡された以下の米国特許出願、2001 年 3 月 22 日出願の米国特許出願第 09/816,004 号「広帯域ネットワークのコンピュータ・アーキテクチャおよびソフトウェア・セル」、2001 年 3 月 22 日出願の米国特許出願第 09/815,554 号「広帯域ネットワーク用のコンピュータ・アーキテクチャのデータ同期システムおよび方法」、2001 年 3 月 22 日出願の米国特許出願第 09/816,020 号「広帯域ネットワーク用のコンピュータ・アーキテクチャのメモリ保護システムおよび方法」、2001 年 3 月 22 日出願の米国特許出願第 09/815,558 号「広帯域ネットワーク用のコンピュータ・アーキテクチャの資源占有システ

ムおよび方法」、および2001年3月22日出願の米国特許出願第09/816,752号「広帯域ネットワーク用のコンピュータ・アーキテクチャの処理モジュール」の一部継続出願であり、その全てをここに援用する。

【背景技術】

【0002】

本発明はコンピュータ・プロセッサ用アーキテクチャとコンピュータ・ネットワークに關し、より詳細には広帯域環境におけるコンピュータ・プロセッサ及びコンピュータ・ネットワーク用アーキテクチャに關する。

【0003】

コンピュータ及び現今のコンピュータ・ネットワークのコンピューティング・デバイス（オフィスのネットワークで使用されるローカル・エリア・ネットワーク（LAN）やインターネットなどのようなグローバルネットワークなど）の計算用デバイスは、スタンド・アローン型の計算用として主として設計されていた。コンピュータ・ネットワークを介するデータとアプリケーション・プログラム（“アプリケーション”）の共用は、これらのコンピュータ及びコンピューティング・デバイスの主要な設計目標ではなかった。これらのコンピュータとコンピューティング・デバイスは、様々な異なるメーカー（モトローラ、インテル、テキサス・インスツルメント、ソニーなど）により製造された広範囲の異なるタイプのプロセッサを用いて一般に設計されたものである。これらのプロセッサの各々はそれ自身の特定の命令セットと命令セット・アーキテクチャ（ISA：instruction set architecture）とを有している。すなわち、それ自身の特定のセットのアセンブリ言語命令と、これらの命令を実行する主演算デバイスと記憶デバイスのための構造とを有する。従って、プログラマは各プロセッサの命令セットとISAとを理解してこれらのプロセッサ用のアプリケーションを書くことを要求される。今日のコンピュータ・ネットワーク上でのコンピュータとコンピューティング・デバイスに異なった種類が混在していることから、データとアプリケーションの共用及びその処理は複雑になっている。さらに、この複数種が混在する環境に対する調整を行うために、多くの場合、同じアプリケーションであっても複数のバージョンを用意することが必要となっている。

【発明の開示】

【発明が解決しようとする課題】

【0004】

グローバルネットワーク、特にインターネットに接続されたタイプのコンピュータやコンピューティング・デバイスは広範囲に及ぶ。パーソナルコンピュータ（PC）やサーバに加えて、これらのコンピューティング・デバイスの中には携帯電話、移動用コンピュータ、個人用情報機器（PDA：personal digital assistant）、セット・トップ・ボックス、デジタルテレビ並びにその他のデバイスが含まれる。コンピュータやコンピューティング・デバイスにおいて異種製品が混在する中でデータやアプリケーションを共用することに起因して、重要な問題が発生している。

【0005】

これらの問題を解決するためのいくつかの手法が試みられてきた。これらの手法の中には特に、優れたインターフェースと複雑なプログラミング手法が含まれる。多くの場合、これらの解決方法では、処理パワーの実質的増加の実現が要求される。また、これらの解決方法では、多くの場合、アプリケーションの処理に必要な時間と、ネットワークを介するデータ伝送に必要な時間とが実質的に増加してしまうという結果が生じる。

【0006】

一般に、データは対応のアプリケーションとは別々に、インターネットを介して伝送される。この手法では、アプリケーションに対応した各セットの伝送データにアプリケーション自体をも送る必要はなくなっている。従ってこの手法により、必要とされる帯域幅の量は最小化されるものの、ユーザには不満の原因となることも多々ある。つまり、クライアント側のコンピュータでは、この伝送データを利用するための適正なアプリケーション、あるいは最新のアプリケーションを入手できない事態も生じうる。またこの手法では、

ネットワーク上のプロセッサによって用いられている複数の異種 I S A と命令セットに対応して、各アプリケーション毎にバージョンの異なる複数のアプリケーションを用意することが要求される。

【0007】

J a v a モデルではこの問題の解決が試みられている。このモデルでは厳しいセキュリティ・プロトコルに準拠する小さなアプリケーション（“アプレット”（applet））が用いられている。アプレットはネットワークを介してサーバー側コンピュータから送信され、クライアント側コンピュータ（“クライアント”）により実行される。異なる I S A を使用しているクライアントごとに、同じアプレットであっても異なるバージョンを送信するという事態を避ける必要があるため、全ての J a v a アプレットはクライアント側の J a v a 仮想マシン上で実行される。J a v a 仮想マシンとは、J a v a I S A と J a v a 命令セットを持つコンピュータをエミュレートするソフトウェアである。しかしながらこのソフトウェアはクライアント側の I S A とクライアント側の命令セットにより実行される。クライアント側では I S A と命令セットが各々異なるが、与えられる J a v a 仮想マシンのバージョンは1つである。従って、複数の各アプレットに異なるバージョンを用意する必要はない。各クライアントでは、当該クライアントにおける I S A と命令セットに対応した適切な J a v a 仮想マシンだけをダウンロードすれば、全ての J a v a アプレットを実行できる。

【0008】

各々の異なる I S A 命令セットに対して異なるバージョンのアプリケーションを書かなければならないという課題は解決されているものの、J a v a の処理モデルでは、クライアント側のコンピュータに対してソフトウェアの追加層が要求される。ソフトウェアのこの追加層のためにプロセッサの処理速度は著しく低下する。この速度の低下は、リアルタイムのマルチメディア・アプリケーションについて特に著しい。また、ダウンロードされた J a v a アプレットの中には、ウイルス、処理上の誤動作などが含まれている可能性がある。これらのウイルスと誤動作は、クライアントのデータベースの破損やその他の損害の原因となる可能性がある。J a v a モデルで用いられているセキュリティ用プロトコルでは、“サンドボックス”（J a v a アプレットがそれ以上はデータを書き込むことができない、クライアント側のメモリ内のスペース）というソフトウェアを設けることによりこの問題の解決が試みられているとはいえ、このソフトウェア駆動型セキュリティ・モデルは多くの場合、その実行時に不安定な状態になり、より多くの処理を必要とする。

【0009】

リアルタイムのマルチメディア・ネットワーク用アプリケーションがますます重要なものになりつつある。これらのネットワーク用アプリケーションは非常に高速な処理が要求される。将来、そのようなアプリケーション用として、毎秒何千メガビットものデータが必要になるかもしれない。ネットワークの現在のアーキテクチャ、及び、特にインターネットのアーキテクチャ、並びに J a v a モデルなどで現在実施されているプログラミング・モデルでこのような処理速度に到達することは非常に難しい。

【0010】

従って、新しいコンピュータ・アーキテクチャと、コンピュータ・ネットワーク用の新しいアーキテクチャと、新しいプログラミング・モデルとが求められている。この新しいアーキテクチャとプログラミング・モデルによって計算上の負担が付加されることなく、ネットワークの様々なメンバー間でのデータとアプリケーションの共用という問題が解決されることが望ましい。この新たなコンピュータ・アーキテクチャとプログラミング・モデルとにより、ネットワークのメンバー間でのアプリケーションとデータの共用時に生じるセキュリティ上の問題も解決されることが望ましい。

【課題を解決するための手段】

【0011】

広帯域ネットワークを介して高速処理を行うコンピュータ・アーキテクチャが提供される。特に、及び、本発明によれば、システムにはデータ通信用のインターフェースデバイ

スと、データ格納用のメモリとが含まれ、メモリには少なくとも1つのメモリ・ロケーションと、それに関連する状態情報とが含まれる。状態情報には第1フィールドとアドレス・フィールドとが含まれ、第1フィールド値が第1の値に等しく、また、アドレス・フィールド値が第2の値に等しい場合、続いて行われる、メモリ・ロケーションに関連付けられたデータの書き込みによって、その中に格納されているデータが、アドレス・フィールド値により示されるアドレスへ書き込まれることになる。

【発明を実施するための最良の形態】

【0012】

本発明の一実施形態においては、システム構成にはプロセッサ・エレメント（PE）と、入力/出力（I/O）インターフェースデバイスと、共用メモリとが含まれる。PEにはさらに、少なくとも1つの処理ユニット（PU）と、1つ以上の付加処理ユニット（APU: attached processing unit）とが含まれる。少なくとも1つのAPUは、I/Oインターフェースデバイスと結合されている外部デバイスからのデータの読出しと外部デバイスへのデータの書き込みを行うことにより、I/O機能を行う。データはAPUとI/Oインターフェースデバイスとの間で、共用メモリを介して、データレベル同期機構を用いて交換される。特に、共用メモリにはデータ格納用の少なくとも1つのメモリ・ロケーションと関連付けられた、少なくとも1つの状態情報ロケーションが含まれ、この状態情報ロケーションには第1フィールドとアドレス・フィールドが含まれる。第1フィールド値が第1の値に等しく、また、アドレス・フィールド値が第2の値に等しい場合、続いて行われる、メモリ・ロケーションに関連付けられたデータの書き込みによって、その中に格納されているデータが、アドレス・フィールド値により示されるアドレスへ書き込まれることになる。

【0013】

図1に、本発明によるコンピュータ・システム101のアーキテクチャ全体を示す。

【0014】

この図に例示されているように、システム101にはネットワーク104が含まれ、複数のコンピュータとコンピューティング・デバイスがこのネットワークと接続されている。ネットワーク104の例として、LAN、インターネットなどのグローバルネットワーク、又は他のコンピュータ・ネットワークが挙げられる。

【0015】

ネットワーク104に接続されているコンピュータとコンピューティング・デバイス（ネットワークの“メンバー”）の中には、クライアント側コンピュータ106、サーバー側コンピュータ108、個人情報機器（PDA: personal digital assistant）、デジタルテレビ（DTV: digital television）112、及びその他の有線又は無線コンピュータとコンピューティング・デバイスなどが含まれる。ネットワーク104のメンバーにより用いられるプロセッサは、同じ共通のコンピューティング・モジュールから構成される。またこれらのプロセッサは、好適には、ISAが全て同じで、同じ命令セットに従って処理を実行する。個々のプロセッサ内に含まれるモジュールの数は、そのプロセッサが必要とする処理パワーにより決められる。

【0016】

例えば、システム101のサーバー108はクライアント106より多いデータ及びアプリケーション処理を実行するので、サーバー108はクライアント106よりも多いコンピューティング・モジュールを含むことになる。一方、PDA110では、最低量の処理しか実行されない。従って、PDA110には最小の数のコンピューティング・モジュールしか含まれない。DTV112はクライアント106とサーバー108間の処理レベルを実行する。従ってDTV112にはクライアント106とサーバー108の間のいくつかのコンピューティング・モジュールが含まれる。以下に解説するように、各コンピューティング・モジュールの中には、処理用コントローラと、ネットワーク104を介して伝送されるデータ及びアプリケーションの並列処理を実行する複数の同一処理ユニットとが含まれる。

10

20

30

40

50

【0017】

システム101がこのような均質な構成を有することから、アダプタビリティ、処理速度、及び処理効率が改善される。システム101の各メンバーが、同じコンピュータ・モジュールの1つ以上（又はコンピュータ・モジュールの一部）を用いて処理を実行するので、データ及びアプリケーションの実際の処理をどのコンピュータ又はコンピュータ・デバイスで実行するかは重要ではなくなる。更に、個々のアプリケーション及びデータの処理は、ネットワークのメンバー間で分担することができる。システム全体を通じて、システム101が処理したデータ及びアプリケーションを含むセルを一意的に識別することにより、この処理がどこで行われたかに関わらず、処理を要求したコンピュータ又はコンピュータ・デバイスへその処理結果を伝送することが可能になる。この処理を実行するモジュールが共通の構造と共通のISAとを有するので、プロセッサ間の互換性を達成するためのソフトウェアの追加層の計算上の負荷が回避される。このアーキテクチャとプログラミング・モデルにより、リアルタイムのマルチメディア・アプリケーションなどの実行に必要な処理速度が改善される。

10

【0018】

システム101により改善された処理速度と効率という利点を更に利用するために、このシステムにより処理されるデータ及びアプリケーションは、一意的に識別される、それぞれフォーマットが同じであるソフトウェア・セル102へとパッケージ化される。各ソフトウェア・セル102はデータ及びアプリケーションの双方を含む、あるいは含み得る。各ソフトウェア・セルはまた、ネットワーク104とシステム101全体の中でセルをグローバルに識別するIDが含まれる。ソフトウェア・セルのこの構造的均一性と、ネットワークの中でのソフトウェア・セルの一意的識別によって、ネットワークの任意のコンピュータ又はコンピュータ・デバイスでのアプリケーションの処理が改善される。例えば、クライアント106はソフトウェア・セル102の作成を行うこともできるが、クライアント106の処理能力は限られていることから、このソフトウェア・セルをサーバー108へ伝送して、処理してもらうこともできる。従って、ソフトウェア・セルはネットワーク104全体を移動してネットワーク上での処理用リソース可用性に基づく処理を行うことが可能となる。

20

【0019】

また、システム101のプロセッサとソフトウェア・セルが均質な構成を有することから、今日の異質なネットワークの混在という問題の多くを防ぐことができる。例えば、任意の命令セットを用いる任意のどのISA上でも、アプリケーションの処理を許容しようとする非効率なプログラミングモデル（Javaの仮想マシンのような仮想マシンなど）が回避される。従って、システム101は今日のネットワークよりもはるかに効果的、かつ、はるかに効率的に、広帯域処理の実現が可能になる。

30

【0020】

ネットワーク104の全てのメンバーのための基本となる処理用モジュールはプロセッサ・エレメント（PE）である。図2にPEの構造を例示する。この図に示すように、PE201は処理ユニット（PU）203、DMAC205、複数の付加処理ユニット（APU）、すなわち、APU207、APU209、APU211、APU213、APU215、APU217、APU219、APU221、を具備する。ローカルPEバス223はAPUと、DMAC205と、PU203との間でデータとアプリケーションを伝送する。ローカルPEバス223は従来のアーキテクチャなどを備えていてもよいし、又はパケット交換式ネットワークとして実装されてもよい。パケット交換式ネットワークとして実装される場合は、より多くのハードウェアが必要となり、その一方で利用可能な帯域幅が増加する。

40

【0021】

PEはデジタル論理回路を実装する様々な方法を用いて構成可能である。しかしながら、PE201は好適にはシリコン基板上の相補型金属酸化膜半導体（CMOS: complementary metal oxide semiconductor）を用いている単一の集積回路として構成される。基

50

板用代替材料の中には、ガリウム砒素、ガリウムアルミニウム砒素、及び多種多様のドーパントを用いるその他のいわゆるIII-V化合物が含まれる。またPE201は超伝導材料（高速単一磁束量子（RSFQ: rapid single-flux-quantum）論理回路など）を用いて実装されることもできる。

【0022】

PE201は高帯域メモリ接続部227を介して、ダイナミック・ランダム・アクセス・メモリ（DRAM）225と密接に関連している。DRAM225はPE201用メイン・メモリとして機能する。DRAM225は好適にはダイナミック・ランダム・アクセス・メモリであることが望ましいといえ、DRAM225は他の手段、例えば、スタティック・ランダム・アクセス・メモリ（SRAM）として、磁気ランダム・アクセス・メモリ（MRAM）、光メモリ又はホログラフィ・メモリなどを用いてDRAM225を実装することも出来る。DMAC205によってDRAM225と、PE201のAPUとPUとの間のデータ伝送が改善される。以下に更に説明するように、DMAC205によって、各APUに対するDRAM225内の排他的領域が指定されるが、この排他的領域の中へはAPUだけしかデータの書き込みができず、また、APUだけしかこの排他的領域からのデータ読出しを行うことができない。また、この排他的領域は“サンドボックス”と呼ばれる。

【0023】

PU203は、データ及びアプリケーションのスタンド・アロン型処理が可能な、標準的プロセッサなどであってよい。作動時に、PU203はAPUによってデータ及びアプリケーションの処理のスケジュール管理と全般的な管理とを行う。APUは好適には単一命令、複数データ（SIMD: single instruction, multiple data）プロセッサであることが望ましい。PU203の制御によって、APUは並列的かつ独立に、これらのデータ及びアプリケーションの処理を実行する。DMAC205は、共用DRAM225に格納されているデータとアプリケーションへのPU203とAPUによるアクセス制御を行う。PE201は好適には8個のAPUを含むことが望ましいといえ、必要とする処理パワーに応じて、PE内での数より多少上下する個数のAPUを用いてもよい。PU203とAPUの一部あるいは全ては、同じハードウェア構造、及び/又は機能を有することが可能である。個々のプロセッサは必要に応じて、ソフトウェアによって制御を行う、又は制御されるプロセッサとして構成される。例えば図3では、PE201は同じアーキテクチャを持つ9個のプロセッサを含むことができる。9個のプロセッサのうちの1つのプロセッサは、制御を行うプロセッサ（例：PU203）と呼ばれる、残りのプロセッサは制御されるプロセッサと呼ばれる（APU207、209、211、213、215、217、219、221など）。また、PE201のようないくつかのPEを結合（まとめてパッケージ化）して処理パワーの改善を図ることができる。

【0024】

例えば、図3に示すように、一つ以上のチップ・パッケージなどの中に4つのPEをパッケージ化してネットワーク104のメンバー用の単一プロセッサを形成してもよい。この構成は広帯域エンジン（BE）と呼ばれる。図3に示すように、BE301には4つのPE（PE303、PE305、PE307、及びPE309）が含まれる。これらのPE間の通信は、BEバス311を介して行われる。広帯域メモリ接続部313によって共用DRAM315とこれらのPE間の通信が行われる。BEバス311の代わりに、BE301のPE間の通信は、DRAM315とこのメモリ接続部を介して行うことができる。

【0025】

入力/出力（I/O）インターフェース317と外部バス319とは、広帯域エンジン301とネットワーク104の他のメンバー間で通信を行う。BE301の各PEは、PEのAPUによって行われるアプリケーションとデータの並列的かつ独立した処理と同様の並列的かつ独立した方法で、データとアプリケーションの処理を実行する。

【0026】

10

20

30

40

50

図4にAPUの構造を例示する。APU402にはローカル・メモリ406、レジスタ410、4つの浮動小数点ユニット412、及び4つの整数演算ユニット414が含まれる。しかし、ここでもまた必要とする処理パワーに応じて、4個より多少上下する個数の浮動小数点演算ユニット412と整数演算ユニット414を用いてもよい。1つの好ましい実施形態では、ローカル・メモリ406には128キロボイトの記憶容量が含まれ、レジスタ410の容量は128×128ビットである。浮動小数点ユニット412は、毎秒320億浮動小数点演算(32GLPOS)で好適に作動し、整数ユニット414は毎秒320億回の演算速度(32GOP)で好適に作動する。

【0027】

ローカル・メモリ402はキャッシュ・メモリではない。ローカル・メモリ402は好適にはSRAMとして構成されることが望ましい。APUに対するキャッシュ・コピーレンシー、つまりキャッシュの整合性のサポートは不要である。PUは当該PUで開始されるダイレクト・メモリ・アクセス(DMA)をサポートするために、キャッシュの整合性が要求される場合もある。しかし、APU又は外部デバイスからの、及び外部デバイスへのアクセスに対するキャッシュの整合性は不要である。

【0028】

APU402には更に、APUへ及びAPUからアプリケーションとデータを伝送するためのバス404が含まれる。1つの好ましい実施形態では、このバスは1024ビットの幅を持つ。APU402には更に内部バス408と、420と、418とが含まれる。1つの好ましい実施形態では、バス408は256ビットの幅を持ち、ローカル・メモリ406とレジスタ410間で通信を行う。バス420と418とは、それぞれ、レジスタ410と浮動小数点演算ユニット412との間、及びレジスタ410と整数ユニット414との間で通信を行う。1つの好ましい実施形態では、レジスタ410から浮動小数点演算ユニット412又は整数演算ユニット414へのバス418の幅は318ビットであり、浮動小数点演算ユニット412又は整数演算ユニット414からレジスタ410へのバス418と420の幅は128ビットである。浮動小数点演算ユニット412又は整数演算ユニット414への幅より広い、レジスタ410から浮動小数点演算ユニットまたは整数演算ユニットへの上記バスの広い幅によって、レジスタ410からのより広いデータ・フローが処理中に許容される。各計算には最大3ワードが必要になる。しかし、各計算結果は一般に1ワードだけである。

【0029】

図5-10は、ネットワーク104のメンバーのプロセッサのモジュラー構造を更に例示する図である。例えば、図5に示すように、1つのプロセッサには単一のPE502を含むことができる。上述のように、このPEには一般に、PUと、DMACと、8個のAPUとが含まれる。各APUにはローカル・ストレージ(LS)が含まれる。一方、プロセッサはビュアライザ(VS)505の構造を有する。図5に示すように、VS505はPU512と、DMAC514と、4つのAPU(APU516、APU118、APU520、APU522)とを有する。PEのその他の4つのAPUによって通常占領されるチップ・パッケージ内のスペースは、この場合、ピクセル・エンジン508、画像用キャッシュ510、及びブラウン管コントローラ(CRTC)504によって占領される。PE502又はVS505に求められる通信速度に応じて、チップ・パッケージの中に光インターフェース506が含まれる場合もある。

【0030】

この標準化されたモジュラー構造を用いて、多数の他のプロセッサが、容易にかつ効率的に構成されうる。例えば、図6に示すプロセッサは、2つのチップ・パッケージ(BE)を備えるチップ・パッケージ602と、4つのVSを含むチップ・パッケージ604とを有する。入出力部(I/O)606によって、チップ・パッケージ602のBEとネットワーク104との間にインターフェースが設けられる。バス608はチップ・パッケージ602とチップ・パッケージ604との間で通信を行う。入出力プロセッサ(IOP)610によってデータ・フローが制御され、I/O606への、またはI/O606からの

10

20

30

40

50

入出力が行われる。1/0606は特定用途向集積回路(ASIC: application specific integrated circuit)として製造が可能である。VSからの出力はビデオ信号612である。

【0031】

図7に、ネットワーク104のその他のメンバーへ超高速通信を行う2つの光インターフェース704と706とを備えたBE702用のチップ・パッケージ(またはローカルに接続された、その他のチップ・パッケージ)を例示する。BE702はネットワーク104のサーバーなどとして機能することができる。

【0032】

図8のチップ・パッケージは2つのPE802と804、及び2つのVS806と808を有する。1/0810はチップ・パッケージとネットワーク104との間にインターフェースを与える。チップ・パッケージからの出力はビデオ信号1である。この構成は画像処理用ワークステーションなどとして機能することができる。

【0033】

図9に更に別の構成を例示する。この構成は、図8に例示されている構成の処理パワーの1/2を含む。2つのPEの代わりに、1つのPE902が設けられ、2つのVSの代わりに1つのVS904が設けられる。1/0906は図8に例示されている1/0の帯域幅の1/2の帯域幅を有する。しかしこのようなプロセッサは、画像処理用ワークステーションとしても機能することができる。

【0034】

最後の構成を図10に示す。このプロセッサは単一のVS1002と1/01004だけから構成される。この構成はPDAなどとして機能することができる。

【0035】

図11Aにネットワーク104のプロセッサのチップ・パッケージの中への光インターフェースの統合を例示する図を示す。これらの光インターフェースによって、光信号は電気信号に変換され、電気信号は光信号へ変換される。また、これらの光インターフェースは、ガリウム砒素、アルミニウムガリウム砒素、ゲルマニウム、及びその他の元素や化合物などを含む様々な材料から構成される。この図に示すように、光インターフェース1104と1106はBE1102のチップ・パッケージ上に組み立てられる。BEバス1108はBE1102のPE、すなわち、PE1110、PE1112、PE1114、PE1116及びこれらの光インターフェースとの間で通信を行う。光インターフェース1104には2つのポート(ポート1118とポート1120)が含まれ、また光インターフェース1106には2つのポート(ポート1122とポート1124)が含まれる。ポート1118、1120、1122及び1124は光導波路1126、1128、1130、1132とそれぞれ接続される。光信号は光インターフェース1104と1106のポートを介して、これらの光導波路の中を通り、BE1102へ、及びBE1102から伝送される。

【0036】

このような光導波路と各BEの4つの光ポートとを用いて、様々な構成において複数のBEをまとめて接続してもよい。例えば図11Bに示すように、このような光ポートを介して2つまたはそれ以上のBE(BE1152、BE1154、BE1156など)を直列に接続することができる。この例では、BE1152の光インターフェース1166は、その光ポートを介してBE1154の光インターフェース1160の光ポートと接続される。同様に、BE1154の光インターフェース1162の光ポートは、BE1156の光インターフェース1164の光ポートと接続される。

【0037】

図11Cにマトリクス構成を例示する。この構成では、各BEの光インターフェースは2つの他のBEと接続される。この図に示すように、BE1172の光インターフェース1188の光ポートの中の1つが、BE1176の光インターフェース1182の光ポートと接続される。光インターフェース1188のもう一方の光ポートは、BE1178の

光インターフェース1184光ポートと接続される。同様に、BE1174の光インターフェース1190の1つの光ポートは、BE1178の光インターフェース1184のもう一方の光ポートと接続される。光インターフェース1190のもう一方の光ポートは、BE1180の光インターフェース1186の光ポートと接続される。このマトリクス構成は他のBEに対して同様に拡張することができる。

【0038】

シリアル構成かマトリクス構成のいずれかを用いて、任意の所望のサイズとパワーからなるネットワーク104用プロセッサ構成が可能になる。言うまでもなく、BEの光インターフェースに対して、あるいはBEよりPE数が上下するプロセッサに対して追加ポートを加えて他の構成を形成してもよい。

【0039】

図12AにBEのDRAMに対する制御システムと構造を例示する。同様の制御システムと構造が、別のサイズを持ち、多少異なる数のPEを含むプロセッサの中で用いられる。この図に示すように、クロスバ交換機によって、BE1201を備える4つのPEからなる各DMAC1210が8つのバンク・コントロール1206と接続される。各バンク・コントロール1206によってDRAM1204の8つのバンク1208（4つだけしか図示されていない）が制御される。したがって、DRAM1204は合計で64のバンクを具備することになる。好ましい実施形態では、DRAM1204は64メガバイトの容量を持ち、各バンクは1メガバイトの容量を持つ。各バンク内の最小のアドレス指定可能単位は、この好ましい実施形態では1024ビットのブロックである。

【0040】

BE1201にはスイッチ・ユニット1212も含まれる。スイッチ・ユニット1212により、BE1201に密接に接続されているBEの他のAPUの、DRAM1204へのアクセスが可能となる。したがって、第2のBEを第1のBEと密接に接続することが可能となり、さらに、各BEの各APUは、APUが通常アクセス可能なメモリ・ロケーション数の2倍のアドレス指定を行うことが可能となる。スイッチ・ユニット1212のようなスイッチ・ユニットを介して、第1のBEのDRAMから第2のBEのDRAMへのデータの直接読み出し、または、第2のBEのDRAMから第1のBEのDRAMへのデータの直接書き込みを行うことが可能となる。

【0041】

例えば、図12Bに示すように、このような書き込みを行うために、第1のBEのAPU（BE1222のAPU1220など）によって、第2のBEのDRAM（通常の場合のようなBE1222のBE1222のDRAM1224ではなく、BE1226のDRAM1228など）のメモリ・ロケーションへの書き込みコマンドが出される。BE1222のDMAC1230は、クロスバ交換機1221を介してバンク・コントロール1234へ書き込みコマンドを送り、バンク・コントロール1234はバンク・コントロール1234と接続された外部ポート1232へコマンドを伝送する。BE1226のDMAC1238は書き込みコマンドを受け取り、このコマンドをBE1226のスイッチ・ユニット1240へ転送する。スイッチ・ユニット1240は書き込みコマンドの中に含まれるDRAMアドレスを識別し、BE1226のバンク・コントロール1242を介して、DRAM1228のバンク1244へ、DRAMアドレス内に格納するデータを送る。したがって、スイッチ・ユニット1240により、DRAM1224とDRAM1228の双方は、BE1222のAPU用の単一メモリ空間として機能することが可能になる。

【0042】

図13にDRAMの64個のバンク構成を図示する。これらのバンクは8つの行（1302、1304、1306、1308、1310、1312、1314、1316）と8つの列（1320、1322、1324、1326、1328、1330、1332、1334）とで構成されている。各行はバンク・コントローラにより制御される。したがって、各バンク・コントローラは8メガバイトのメモリを制御する。

【0043】

10

20

30

40

50

図14Aと14Bに最小のアドレス指定可能な格納単位(1024ビットのブロックなど)でのDRAMの格納とアクセスを行うための異なる構成を例示する。図14Aでは、DMAC1402は単一のバンク1404の中に8つの1024ビット・ブロック1406を格納する。一方で図14Bでは、DMAC1412によって、1024ビットを含むデータ・ブロックの読み出しと書き込みが行われるが、これらのブロックは2つのバンク(バンク1414とバンク1416)の間で分配される。したがって、これらのバンクの各々には16個のデータ・ブロックが含まれ、データの各ブロックには512ビットが含まれる。この分配によって、DRAMのアクセスをさらに高速なものに改善することが可能となり、ある種のアプリケーションの処理に役立つ。

【0044】

図15にPE内のDMAC1504のアーキテクチャを例示する。この図に示されているように、各APUがDMAC1506の構造上のノード1504へ直接アクセスを行うように、DMAC1506を含む構造上のハードウェアはPEを通じて配設される。各ノードは、ノードが直接アクセスを行う対象のAPUによるメモリ・アクセスに適した論理処理を実行する。

【0045】

図16にDMACの他の実施形態、すなわち、非分散型アーキテクチャを例示する。この場合、DMAC1606の構造上のハードウェアは集中型である。APU1602とPU1604は、ローカルPEバス1607を介してDMAC1606を用いて通信を行う。DMAC1606はクロスバ交換機を介して、バス1608と接続される。バス1608はDRAM1610と接続されている。

【0046】

上述のように1つのPEの複数のAPUの全ては、独立に、共用DRAM内のデータへのアクセスが可能である。その結果、第1のAPUがあるデータをそのローカル・ストレージで処理しているときに、第2のAPUがこれらのデータを要求する場合もある。その時点で共用DRAMから第2のAPUへデータが出力された場合、データの値を変化させる第1のAPUの進行中の処理に起因して、そのデータが無効になる場合がある。したがって、その時点で第2のプロセッサが共用DRAMからデータを受け取った場合、第2のプロセッサでエラー結果が生じる恐れがある。例えば、このようなデータとしては、グローバル変数用の具体的な値が上げられる。第1のプロセッサがその処理中にその値を変えた場合、第2のプロセッサはもう使用されていない値を受け取ることになる。したがって、共用DRAMの範囲内で、メモリ・ロケーションからの、及び、メモリ・ロケーションへのAPUによるデータの読み出しと書き込みを同期させる何らかの方式が必要となる。この方式では、別のAPUがそのローカル・ストレージで現在備わっている対象データであって、したがって最新のものではないデータのメモリ・ロケーションからの読み出しと、最新のデータを格納するメモリ・ロケーションの中へのデータの書き込みと、を行わないようにする必要がある。

【0047】

これらの問題を解決するために、DRAMの各アドレス指定が可能なメモリ・ロケーションに対して、そのメモリ・ロケーションに格納されているデータに関連する状態情報を格納するために、DRAMの中でメモリの追加セグメントの割り振りが行われる。この状態情報の中には、フル・エンプティ(F/E)ビットと、メモリ・ロケーションからデータを要求するAPUの識別子(APUID)と、要求されたデータを読み出す読み出し先となるAPUのローカル・ストレージのアドレス(LSアドレス)とが含まれる。DRAMのアドレス指定可能なメモリ・ロケーションは任意のサイズとすることができる。ある好ましい実施形態では、このサイズは1024ビットである。

【0048】

F/Eビットの1への設定は、関連するメモリ・ロケーションに格納されているデータが最新のものであることを示す。一方、F/Eビットの0への設定は、関連するメモリ・ロケーションに格納されているデータが最新のものではないことを示す。このビットが0

10

20

30

40

50

に設定されているときにAPUがデータを要求しても、APUによってそのデータの即時読み出しは妨げられる。この場合、そのデータを要求しているAPUを識別するAPUIDと、データが最新のものになっているとき、そのデータを読み出す読み出し先となるこのAPUローカル・ストレージ内のメモリ・ロケーションを識別するLSアドレスとが、追加メモリ・セグメントの中へ入力される。

【0049】

また追加メモリ・セグメントは、APUのローカル・ストレージ内の各メモリ・ロケーションに対して割り振られる。この追加メモリ・セグメントは“ビジー・ビット”と呼ばれる1ビットを格納する。ビジー・ビットはDRAMから検索される固有データの格納用として、関連するLSメモリ・ロケーションの予約を行うために使用される。ローカル・ストレージの特定のメモリ・ロケーションに対してビジー・ビットが1に設定されている場合、これらの固有データの書き込み用としてのみ、APUはこのメモリ・ロケーションを使用することができる。一方、ローカル・ストレージの特定のメモリ・ロケーションに対して、ビジー・ビットが0に設定されている場合、APUは任意のデータの書き込み用にこのメモリ・ロケーションを使用することができる。

【0050】

F/Eビット、APUID、LSアドレス、及びビジー・ビットが、PEの共用DRAMからの、及び、PEの共用DRAMへのデータの読み出しと書き込みを同期させるために使用される方法の例を、図17Aから図17Oに例示する。

【0051】

図17Aに示すように、1つ以上のPE（PE1720など）がDRAM1702を使用する。PE1720にはAPU1722とAPU1740が含まれる。APU1722には制御論理回路1724が含まれ、APU1740には制御論理回路1742が含まれる。APU1722にはローカル・ストレージ1726も含まれる。このローカル・ストレージには複数のアドレス可能なメモリ・ロケーション1728が含まれる。APU1740にはローカル・ストレージ1744が含まれ、このローカル・ストレージにも複数のアドレス可能なメモリ・ロケーション1746が含まれる。これらのアドレス可能なメモリ・ロケーションのすべては、好適にはサイズが1024ビットであることが望ましい。

【0052】

メモリの追加セグメントは各LSのアドレス可能なメモリ・ロケーションと関連付けられる。例えば、メモリ・セグメント1729と1734とはそれぞれ、ローカルなメモリ・ロケーション1731と1732と関連付けられ、メモリ・セグメント1752はローカル・メモリ・ロケーション1750と関連付けられる。上述のような“ビジー・ビット”はこれらの追加のメモリ・セグメントの各々の中に格納される。ローカル・メモリ・ロケーション1732は、このメモリ・ロケーションがデータを含むことを示すいくつかのX印を用いて示されている。

【0053】

DRAM1702には、メモリ・ロケーション1706と1708を含む、複数のアドレス可能なメモリ・ロケーション1704が含まれる。またこれらのメモリ・ロケーションは、好適にはサイズが1024ビットであることが望ましい。メモリの追加セグメントはまた、これらのメモリ・ロケーションの各々とも関連付けられる。例えば、追加のメモリ・セグメント1760はメモリ・ロケーション1706と関連付けられ、また、追加のメモリ・セグメント1762はメモリ・ロケーション1708と関連付けられる。各メモリ・ロケーションに格納されるデータに関連する状態情報は、メモリ・ロケーションに関連付けられたメモリに格納される。この状態情報の中には、上述のように、F/Eビット、APUID、及びLSアドレスが含まれる。例えば、メモリ・ロケーション1708については、この状態情報には、F/Eビット1712、APUID1714、及びLSアドレス1716が含まれる。

【0054】

10

20

30

40

50

この状態情報とビジー・ビットとを用いて、PEのAPU、又は1グループのPE間での、共用DRAMからの、および、同期した共用DRAMからの読み出しと、同期した共用DRAMへのデータの書き込みを行うことができる。

【0055】

図17Bに、APU1722のLSメモリ・ロケーション1732から、DRAM1702のメモリ・ロケーション1708へのデータの同期書き込みの開始を例示する図を示す。APU1722の制御論理回路1724によって、これらのデータの同期書き込みが開始される。メモリ・ロケーション1708はエンプティであるため、F/Eビット1712は0に設定される。その結果、メモリ・ロケーション1708の中へLSロケーション1732内のデータを書き込むことが可能となる。このビットが1に設定され、メモリ・ロケーション1708がフル状態であり、最新の有効データを含むことが示されている場合、制御回路1724はエラー・メッセージを受け取ることになり、このメモリ・ロケーションへのデータの書き込みは禁止される。

【0056】

メモリ・ロケーション1708への成功したデータの同期書き込みの結果を図17Cに示す。この書き込まれたデータはメモリ・ロケーション1708に格納され、F/Eビット1712は1に設定される。この設定により、メモリ・ロケーション1708がフル状態であること、及び、このメモリ・ロケーションの中のデータが最新であり有効であることが示される。

【0057】

図17Dに、DRAM1702のメモリ・ロケーション1708から、ローカル・ストレージ1744のLSメモリ・ロケーション1750へのデータの同期読み出しの開始を例示する図を示す。この読み出しを開始するために、LSメモリ・ロケーション1750のメモリ・セグメント1752の中のビジー・ビットが1に設定されて、このデータに対するメモリ・ロケーションが予約される。このビジー・ビットを1に設定することにより、APU1740がこのメモリ・ロケーションにその他のデータを格納することはなくなっている。

【0058】

図17Eに示すように、制御論理回路1742は次に、DRAM1702のメモリ・ロケーション1708に対し、同期読取りコマンドを出す。このメモリ・ロケーションに関連するF/Eビット1712は1に設定されているので、メモリ・ロケーション1708に格納されるデータは最新の有効データであると見なされる。その結果、メモリ・ロケーション1708からLSメモリ・ロケーション1750へのデータ転送の準備の際に、F/Eビット1712は0に設定される。この設定は図17Fに示されている。このビットを0に設定することは、これらのデータの読み出しの後に、メモリ・ロケーション1708のデータは無効になることを示す。

【0059】

図17Gに示すように、メモリ・ロケーション1708内のデータは次に、メモリ・ロケーション1708からLSメモリ・ロケーション1750へ読み出される。図17Hは最終状態を示す。メモリ・ロケーション1708のデータのコピーはLSメモリ・ロケーションに格納される。F/Eビット1712は0に設定され、メモリ・ロケーション1708のデータは無効であることが示される。この無効はAPU1740によって行われた上記データの変更の結果である。メモリ・セグメント1752のビジー・ビットも0に設定される。この設定によって、APU1740がLSメモリ・ロケーション1750を任意の目的に利用できること、すなわち、このLSメモリ・ロケーションがもはや固有データの受信を待機している予約状態ではないことが示される。したがって、LSメモリ・ロケーション1750は任意の目的のために、APU1740によるLSメモリ・ロケーション1750へのアクセスが可能になる。

【0060】

図17Iから17Oに、DRAM1702のメモリ・ロケーション用F/Eビットが0

10

20

30

40

50

に設定され、このメモリ・ロケーションのデータが最新のものでもなく有効なものでもないことが示されている場合の、DRAM 1702（メモリ・ロケーション1708など）のメモリ・ロケーションから、APUのローカル・ストレージ（ローカル・ストレージ1744のLSメモリ・ロケーション1752など）のLSメモリ・ロケーションへのデータの同期読み出しが例示されている。図17Iに示すように、この転送を開始するために、LSメモリ・ロケーション1750のメモリ・セグメント1752内のビジー・ビットは1に設定され、このデータ転送用としてこのLSメモリ・ロケーションが予約される。図17Jに示すように、制御論理回路1742は次に、DRAM 1702のメモリ・ロケーション1708に対し、同期読取りコマンドを出す。このメモリ・ロケーションと関連付けられたF/Eビット（F/Eビット1712）は0に設定されているので、メモリ・ロケーション1708に格納されているデータは無効である。その結果、信号は制御論理回路1742へ伝送され、このメモリ・ロケーションからのデータの即時読み出しが阻止される。

【0061】

図17Kに示すように、APU ID1714とこの読取コマンド用のLSアドレス1716は次にメモリ・セグメントの中へ書き込まれる。この場合、APU1740用のAPU IDとLSメモリ・ロケーション1750用のLSメモリ・ロケーションは、メモリ・セグメント1762の中へ書き込まれる。したがって、メモリ・ロケーション1708内のデータが最新のものになっているとき、このAPU IDとLSメモリ・ロケーションは、最新のデータを伝送する伝送先のメモリ・ロケーションを決定するために使用される。

【0062】

メモリ・ロケーション1708内のデータは、APUがこのメモリ・ロケーションの中へデータを書き込むと、有効で最新のデータとなる。APU1722のメモリ・ロケーション1732などから、メモリ・ロケーション1708への、データの同期書き込みが図17Lに示されている。このメモリ・ロケーション用のF/Eビット1712が0に設定されているため、これらのデータのこの同期書き込みは許される。

【0063】

図17Mに示すように、この書き込み後、メモリ・ロケーション1708のデータは最新の有効データになる。したがって、メモリ・セグメント1762から得られるAPU ID1714とLSアドレス1716とは、メモリ・セグメント1762から即座に読み出され、次いで、この情報はこのセグメントから削除される。メモリ・ロケーション1708のデータの即時読み出しを予測して、F/Eビット1712もまた0に設定される。図17Nに示すように、APU ID1714とLSアドレス1716とを読み出すと、APU1740のLSメモリ・ロケーション1750へメモリ・ロケーション1708内の有効データを読み出すために、この情報は直ちに使用される。図17Oに最終状態を示す。この図は、メモリ・ロケーション1708からメモリ・ロケーション1750にコピーされた有効データと、0に設定されたメモリ・セグメント1752内のビジー・ビットと、0に設定されたメモリ・セグメント1762内のF/Eビット1712とを示す。このビジー・ビットを0に設定することにより、任意の目的のためにAPU1740がLSメモリ・ロケーション1750のアクセスを行うことができる。このF/Eビットを0に設定することにより、メモリ・ロケーション1708内のデータがもはや最新のものではなく、有効なものでもないことが示される。

【0064】

図18は上述のオペレーションと、DRAMのメモリ・ロケーションの様々な状態とを要約する図であり、この状態はF/Eビットの状態と、APU IDと、メモリ・ロケーションに対応するメモリ・セグメントの中に格納されたLSアドレスとに基づく。このメモリ・ロケーションは、3つの状態を持つことが可能である。これらの3つの状態として、F/Eビットが0に設定され、APU IDまたはLSアドレスに対して情報が提供されないエンプティ状態1880と、F/Eビットが1に設定され、APU IDまたはL

10

20

30

40

50

S アドレスに対して情報が提供されないフル状態 1882 と、F/E ビットが 0 に設定され、A P U ID と L S アドレスに対して情報が提供されるブロッキング状態 1884 とがある。

【0065】

この図に示すように、エンプティ状態 1880 では、同期書き込みオペレーションが許され、フル状態 1882 への遷移という結果が得られる。しかし、メモリ・ロケーションがエンプティ状態であるときは、メモリ・ロケーション内のデータが最新のものではないので、同期読み出しオペレーションに対しては、ブロッキング状態へ遷移するという結果となる。

【0066】

フル状態 1882 では、同期読み出しオペレーションが許され、エンプティ状態 1880 への遷移という結果が得られる。一方、有効データの上書きを避けるために、フル状態 1882 の同期書き込みオペレーションは禁止される。このような書き込みオペレーションがこの状態で試みられる場合、状態の変化は生じず、エラー・メッセージが A P U の対応する制御論理回路へ伝送される。

【0067】

ブロッキング状態 1884 では、メモリ・ロケーションの中へのデータの同期書き込みが許され、エンプティ状態 1880 への遷移という結果が得られる。一方、ブロッキング状態 1884 での同期読み出しオペレーションは禁止される。それは、このブロッキング状態を生じさせることとなった前回の同期読み出しオペレーションとのコンフリクトを阻止するためである。同期読み出しオペレーションがブロッキング状態 1884 で試みられた場合、状態変化は生じず、A P U の対応する論理制御回路へエラー・メッセージが伝送される。

【0068】

共用 D R A M からのデータの同期読み出しと、共用 D R A M へのデータの同期書き込みを行う上述の方式は、外部デバイスからのデータ読み出しと、外部デバイスのデータ書き込み用プロセッサとして通常専用の計算用リソースを省くためにも利用することができる。この入力/出力 (I/O) 機能は P U によって行うこともできる。しかし、この同期方式の変更を利用して、適切なプログラムを実行する A P U がこの機能を実行してもよい。例えば、この方式を利用して、外部デバイスによって開始された I/O インターフェースからのデータ伝送を求める割込み要求を受け取る P U は、この A P U にこの要求処理を委任してもよい。次に A P U は、I/O インターフェースに対して同期書き込みコマンドを出す。今度はこのインターフェースによって、現在 D R A M の中へデータを書き込むことができる旨の信号が外部デバイスへ送られる。次に A P U は D R A M へ同期読み取りコマンドを出し、D R A M の関連するメモリ空間をブロッキング状態に設定する。A P U はまた、データを受け取る必要がある A P U のローカル・ストレージのメモリ・ロケーションに対して、ビジー・ビットを 1 に設定する。ブロッキング状態では、D R A M の関連するメモリ空間と関連付けられた追加メモリ・セグメントの中に、A P U の ID と A P U のローカル・ストレージの関連するメモリ・ロケーションのアドレスが含まれる。次に外部デバイスは同期書き込みコマンドを出し、D R A M の関連するメモリ空間へデータが直接書き込まれる。このメモリ空間はブロッキング状態にあるので、データは、この空間の中から、追加メモリ・セグメントの中で識別された A P U のローカル・ストレージのメモリ・ロケーションの中へ直ちに読み出される。次いで、これらのメモリ・ロケーション用のビジー・ビットは 0 に設定される。外部デバイスがデータの書き込みを完了したとき、A P U は伝送が完了した旨を示す信号を P U へ出す。

【0069】

したがって、この方式を用いて、P U に対する最小の計算上の負荷で、外部デバイスからのデータ転送処理を行うことができる。しかし、この機能を任された A P U は P U に対して割込み要求を出せることが望ましく、外部デバイスが D R A M に対して直接アクセスを行うことが望ましい。

【0070】

各PEのDRAMには複数の“サンドボックス”が含まれる。サンドボックスによって共用DRAM領域が判定され、この領域を超えて、特定のAPUまたは1組のAPUがデータの読み出しや書き込みを行うことはできない。これらのサンドボックスによって、1つのAPUが処理するデータに起因する、別のAPUによって処理されるデータの破損に対するセキュリティが与えられる。またこれらのサンドボックスによって、ソフトウェア・セルが全DRAMの中でデータの破損を生じる可能性はなく、ネットワーク104から特定のサンドボックスの中へソフトウェア・セルのダウンロードを行うことが許される。本発明では、サンドボックスはDRAMとDMACとからなるハードウェアの中に設けられる。ソフトウェアの代わりに、このハードウェア内にこれらのサンドボックスを設けることにより、速度とセキュリティという利点が得られる。

10

【0071】

PEのPUはAPUへ割り当てられるサンドボックスの制御を行う。PUは通常、オペレーティング・システムのような信頼のおけるプログラムだけしか動作させないので、この方式によって、セキュリティが危険にさらされることはない。この方式に従って、PUはキー管理テーブルの構築と維持を行う。図19にこのキー管理テーブルを例示する。この図に示すように、キー管理テーブル1902の各エントリには、APU用の識別子(ID)1904と、そのAPU用のAPUキー1906と、キー・マスク1908とが含まれる。このキー・マスクの用途について以下に説明する。キー管理テーブル1902は好適にはスタティック・ランダム・アクセス・メモリ(SRAM)のような比較的高速のメモリに格納され、DMACと関連付けられる。キー管理テーブル1902へのエントリはPUによって制御される。APUがDRAMの特定格納位置(ストレージロケーション)へのデータの書き込みとあるいはDRAMの特定の格納位置からのデータの読み出しを要求すると、DMACはその格納位置と関連付けられたメモリ・アクセス・キーに対してキー管理テーブル1902内のそのAPUへ割り当てられたAPUキー1906の評価を行う。

20

【0072】

図20に示すように、DRAM2002の各アドレス可能な格納位置2006に対して専用メモリ・セグメント2010が割り当てられる。この格納位置用のメモリ・アクセス・キー2012はこの専用メモリ・セグメントの中に格納される。上述のように、やはり各アドレス可能格納位置2006と関連付けられたさらなる追加専用メモリ・セグメント2008によって、格納位置へのデータを書き込みと、格納位置からのデータ読み出しを行うための同期情報が格納される。

30

【0073】

作動時に、APUはDMACへDMAコマンドを出す。このコマンドにはDRAM2002の格納位置2006のアドレスが含まれる。このコマンドを実行する前に、DMACは、キー管理テーブル1902におけるAPUのID1904を用いて要求を行っているAPUのキー1906を調べる。次いでDMACはAPUがアクセスを求める対象先であるDRAMの格納位置と関連付けられた専用メモリ・セグメント2010の中に格納されるメモリ・アクセス・キー2012と、要求を行っているAPUのAPUキー1906との比較を行う。2つのキーが一致しない場合、DMAコマンドは実行されない。一方、2つのキーが一致する場合、DMAコマンドは進行し、要求されたメモリ・アクセスが実行される。

40

【0074】

図21に他の実施形態を例示する。この実施形態では、PUはメモリ・アクセス管理テーブル2102の維持も行う。メモリ・アクセス管理テーブル2102にはDRAM内にある各サンドボックス用のエントリが含まれる。図21の特定の例では、DRAMには64個のサンドボックスが含まれる。メモリ・アクセス管理テーブル2102内の各エントリには、サンドボックス用識別子(ID)2104と、ベース・メモリ・アドレス2106と、サンドボックス・サイズ2108と、メモリ・アクセス・キー2110と、アクセ

50

ス・キー・マスク 2112 とが含まれる。ベース・メモリ・アドレス 2106 によって、DRAM にアドレスが設けられこのアドレスによって特定のメモリ・サンドボックスの最初の部分が示される。サンドボックス・サイズ 2108 によりサンドボックスのサイズが与えられ、したがって、このサイズにより特定のサンドボックスのエンドポイントが与えられる。

【0075】

図 22 はキー管理テーブル 1902 とメモリ・アクセス管理テーブル 2102 とを用いて DMA コマンドを実行するためのステップを示すフローチャートである。ステップ 2202 では、APU によってサンドボックス内の特定 1 つあるいは複数のメモリ・ロケーションに対するアクセス用 DMA コマンドが DMAC へ出される。このコマンドにはアクセス要求を行う対象先である特定のサンドボックスの識別を行うサンドボックス ID 2104 が含まれる。ステップ 2204 で、DMAC は APU の ID 1904 を利用して、キー管理テーブル 1902 内の要求を行っている APU のキー 1906 を調べる。ステップ 2206 で、DMAC はメモリ・アクセス管理テーブル 2102 で、サンドボックスと関連付けられたメモリ・アクセス・キー 2110 を調べるコマンドで、サンドボックス ID 2104 を利用する。ステップ 2208 で、DMAC は、要求を行っている APU へ割り当てられている APU キー 1906 と、サンドボックスと関連付けられたアクセス・キー 2110 と比較する。ステップ 2110 で、2 つのキーが一致するかどうかの決定が行われる。2 つのキーが一致しない場合、処理はステップ 2212 へ移行し、そこで DMA コマンドは先へ進まず、要求を行っている APU と P U のいずれか、または双方へエラーメッセージが送信される。一方、ステップ 2210 で、2 つのキーの一致が得られた場合、処理はステップ 2214 へと進み、そこで DMAC は DMA コマンドを実行する。

【0076】

APU 用キーとメモリ・アクセス・キー用のキー・マスクにより、このシステムに大きな柔軟性が与えられる。キー用のキー・マスクにより、マスクされたビットはワイルド・カードに変換される。例えば、APU キー 1906 と関連付けられたキー・マスク 1908 が、キー・マスク 1908 内のこれらのビットを 1 に設定することなどにより、その最後の 2 ビットが“マスク”に設定されている場合、APU キーは 1 又は 0 のいずれかになることができ、そのままメモリ・アクセス・キーに一致することになる。例えば、APU キーは 1010 であるとする。通常、この APU キーによって 1010 のアクセス・キーを持つサンドボックスへのアクセスだけが可能になる。しかし、この APU キー用の APU キー・マスクが 0001 に設定されている場合、この APU キーを用いて 1010 または 1011 のいずれかのアクセス・キーを持つサンドボックスへのアクセスを行うことが可能となる。同様に、1010 または 1011 のいずれかの APU キーを持つ APU によって、0001 に設定されたマスクを持つアクセス・キー 1010 のアクセスを行うことが可能である。APU キー・マスクとメモリ・キーマスクの双方を同時に使用することができるので、多数のバリエーションのサンドボックスに対する APU によるアクセシビリティの設定が可能となる。

【0077】

本発明はまた、システム 101 のプロセッサ用の新しいプログラミング・モデルも提供する。このプログラミング・モデルではソフトウェア・セル 102 が用いられる。ネットワーク 104 上の任意のプロセッサへ処理用として、これらのセルの伝送を行うことが可能である。また、この新しいプログラミング・モデルでは、システム 101 のユニークなモジュール形アーキテクチャとシステム 101 のプロセッサとが利用される。

【0078】

ソフトウェア・セルは APU のローカル・ストレージから APU によって直接処理される。APU は DRAM 内のいずれのデータ又はプログラムに対しても直接働きかけことは行わない。DRAM 内のデータとプログラムは、APU がこれらのデータとプログラムの処理を行う前に、APU のローカル・ストレージの中に読み込まれる。したがって、APU のローカル・ストレージには、プログラム・カウンタと、スタックと、これらのプロ

10

20

30

40

50

グラムを実行するための他のソフトウェア・エレメントが含まれる。PUはDMACに対してDMACコマンドを出すことによりAPUの制御を行う。

【0079】

図23にソフトウェア・セル102の構造を例示する。この図に示すように、ソフトウェア・セル2302などのソフトウェア・セルの中には、ルート選定情報セクション2304と本体部分2306とが含まれる。ルート選定情報セクション2304に含まれる情報は、ネットワーク104のプロトコルに依って決められる。ルート選定情報セクション2304にはヘッダ2308、宛先ID2310、ソースID2312、及び応答ID2314が含まれる。宛先ID2310にはネットワーク・アドレスが含まれる。TCP/IPプロトコルの下で、例えば、ネットワーク・アドレスはインターネット・プロトコル(I/P)である。更に、宛先ID2310には、処理のためにセルを送送すべき伝送先のPE及びAPUの識別子が含まれる。ソースID2314にはネットワーク・アドレスが含まれ、このソースIDによってPEとAPUとが識別され、このPEとAPUとからセルが起動し、必要な場合に、宛先PEとAPUとがセルに関する追加情報を得ることが可能となる。応答ID2314にはネットワーク・アドレスが含まれ、この応答ID2314によって、セルに関するクエリとセルの処理結果とを送る送り先のPEとAPUとが識別される。

【0080】

セルの本体部分2306にはネットワークのプロトコルとは無関係の情報が含まれる。図23の分解部分は、セルの本体部分2306の細部を示す。セルの本体部分2306のヘッダ2320によってセル本体部分の開始部が識別される。セル・インターフェース2322にはセルの利用に必要な情報が含まれる。この情報にはグローバルな一意のID2324と要求されるAPU2326と、サンドボックス・サイズ2328と、前回のセルのID2330が含まれる。

【0081】

グローバルな一意のID2324はネットワーク104全体を通じてソフトウェア・セル2302を一意的に識別する。グローバルな一意のID2324はソースID2312(ソースID2312内のPE又はAPUの一意的識別子など)と、ソフトウェア・セル2302の作成又は伝送の時刻と日付に基づき作成される。必要なAPU2326によってセルの実行に必要な最低数のAPUが与えられる。サンドボックス・サイズ2328により、セルの実行に必要なDRAMと関連する必要なAPU内に、保護されたメモリ量が与えられる。前回のセルID2330により、シーケンシャルな実行を要求する1グループのセル(ストリーミング・データなど)内の前回のセルの識別子が提供される。

【0082】

実行セクション2332にはセルのコア情報が含まれる。この情報には、DMAコマンド・リスト2334と、プログラム2336と、データ2338とが含まれる。プログラム2336には、APUプログラム2360や2338などの、APUによって実行されるプログラム(“アプレット”と呼ばれる)が含まれ、データ2338にはこれらのプログラムを用いて処理されるデータが含まれる。DMAコマンド・リスト2334には、プログラムの起動に必要な一連のDMAコマンドが含まれる。これらのDMAコマンドには、DMAコマンド2340、2350、2355、2358が含まれる。PUはDMACへこれらのDMAコマンドを出す。

【0083】

DMAコマンド2340にはVID2342が含まれる。VID2342は、DMAコマンドが出されたときに、物理IDに対して対応付けられるAPUのバーチャルIDである。DMAコマンド2340には、ロード・コマンド2344とアドレス2346も含まれる。ロード・コマンド2344はAPUにDRAMから特定の情報を読み出して、ローカル・ストレージの中へ入れるように命令する。アドレス2346によって、この特定情報を含むDRAM内のバーチャル・アドレスが与えられる。この特定情報は、プログラム・セクション2336からのプログラム、データ・セクション2338からのデータ、又

はその他のデータなどであってよい。最終的に、DMAコマンド2340にはローカル・ストレージのアドレス2348が含まれる。このアドレスにより、情報をロードできそうなローカル・ストレージのアドレスが識別される。DMAコマンド2350には同様の情報が含まれる。その他のコマンドも使用可能である。

【0084】

DMAコマンド・リスト2334には、一連のキック・コマンド(キック・コマンド2355や2358など)も含まれる。キック・コマンドとは、PUによってAPUへ出される、セルの処理を開始するコマンドである。DMAキック・コマンド2355にはバーチャル・APU ID2352と、キック・コマンド2354と、プログラム・カウンタ2356とが含まれる。バーチャルAPU ID2352はキックすべき対象APUを識別し、キック・コマンド2354は関連するキック・コマンドを与え、プログラム・カウンタ2356はプログラムの実行用プログラム・カウンタのためのアドレスを与える。DMAキック・コマンド2358は同じAPU又は別のAPUに対して同様の情報を与える。

【0085】

上述のように、PUは独立したプロセッサとしてAPUを扱い、コプロセッサとして扱うものではない。したがって、APUによる処理を制御するために、PUは遠隔手順呼出しに類似したコマンドを使用する。これらのコマンドは“APU遠隔手順呼出し(ARPC)”と呼ばれる。PUは一連のDMAコマンドをDMACへ出すことにより、ARPCを実行する。DMACはAPUプログラムとそれに関連するスタック・フレームとをAPUのローカル・ストレージへロードする。次いでPUはAPUへ最初のキックを出し、APUプログラムを実行する。

【0086】

図24にアプレットを実行するためのARPCのステップを例示する。指定のAPUによるアプレットの処理の開始時にPUが実行するこれらのステップが、図24の第1の部分2404に示されている。

【0087】

ステップ2410で、PUはアプレットを評価し、次にアプレットの処理用APUを指定する。ステップ2412で、PUは、必要な単体のサンドボックス用のメモリ・アクセス・キーの設定を行うDMAコマンドをDMACへ出すことにより、アプレットの実行用スペースをDRAM内に割り振る。ステップ2414で、PUは指定APUへの割込み要求による、アプレットの完了信号の伝送を可能にする。ステップ2418で、PUはDRAMからAPUのローカル・ストレージへアプレットをロードするDMAコマンドをDMACへ出す。ステップ2420で、DMAコマンドが実行され、DRAMからAPUのローカル・ストレージへアプレットが読み出される。ステップ2422で、PUは、アプレットと関連付けられたスタック・フレームをDRAMからAPUのローカル・ストレージへロードするDMAコマンドをDMACへ出す。ステップ2423で、DMAコマンドが実行され、スタック・フレームがDRAMからAPUのローカル・ストレージへ読み出される。ステップ2424で、PUは、DMACがAPUへキーを割り当てて、ステップ2412で指定された、一又は複数のハードウェア・サンドボックスからのデータ読み出しと、その一又は複数のハードウェア・サンドボックスへのデータ書き込みを行うことをAPUに許可するDMAコマンドを出す。ステップ2426で、DMACはAPUへ割り当てられたキーを用いて、キー管理テーブル(KTAB)の更新を行う。ステップ2428で、PUは、プログラムの処理を開始するDMAコマンド“キック”をAPUに出す。特定のアプレットに応じて、特定のARPCの実行時に、PUによって他のDMAコマンドを出してもよい。

【0088】

上述のように、図24の第2の部分2404はアプレットの実行時にAPUにより行われるステップを例示するものである。ステップ2430で、APUは、ステップ2428で出されるキック・コマンドに応じてアプレットの実行を開始する。ステップ2432で

、アプレットの指示で、APUはアプレットの関連スタック・フレームの評価を行う。ステップ2434で、APUはDMACへ複数のDMAコマンドを出し、スタック・フレームが必要に応じてDRAMからAPUのローカル・ストレージへ指定するデータのロードを行う。ステップ2436で、これらのDMAコマンドが実行され、データはDRAMからAPUのローカル・ストレージへ読み出される。ステップ2438で、APUはアプレットを実行し、ある結果を出力する。ステップ2440で、APUはDMACへDMAコマンドを出し、DRAMにその結果を格納する。ステップ2442で、DMAコマンドが実行され、アプレットの結果がAPUのローカル・ストレージからDRAMへ書き込まれる。ステップ2444で、APUはPUへ割込み要求を出し、ARPCが完了したことを示す信号伝送を行う。

【0089】

PUの指示の下で独立にタスクを実行するAPUの能力によって、1グループのAPUと、1グループのAPUと関連付けられたメモリ・リソースとを拡張タスクの実行専用にすることが可能になる。例えば、1つのPUは、1以上のAPUと、これらの1以上のAPUと関連付けられた1グループのメモリ・サンドボックスとを、拡張された時間中、ネットワークを介して伝送されてくるデータの受信専用とし、また、1以上の他のAPUとそれらと関連付けられたメモリ・サンドボックスへ、この時間中受信したデータの更なる処理を行うための送信専用とすることができる。この能力は、ネットワーク104を介して伝送されるストリーミングデータ（ストリーミングMPEG又はストリーミングATRACオーディオ又はビデオ・データなど）の処理によって特に好適である。PUは、1以上のAPU及びそれらと関連付けられたメモリ・サンドボックスをこれらのデータの受信専用とし、1以上の他のAPU及びそれらと関連付けられたメモリ・サンドボックスをこれらのデータの解凍と処理専用とすることができる。言い換えれば、PUはAPUのグループとそれらと関連付けられたメモリ・サンドボックスとの間でこのようなデータ処理を行うための専用パイプライン関係の確立を行うことができる。

【0090】

しかし、このような処理を効率的に実行するためには、パイプラインの専用APUとメモリ・サンドボックスとが、データ・ストリームを含むアプレットの処理が行われない時間中もパイプライン専用のままであることが望ましい。言い換えれば、専用APUとその関連付けられたサンドボックスが、これらの時間中、予約状態のままに置かれることが望ましい。アプレットの処理の完了時における、APUとその関連付けられた又は複数のメモリ・サンドボックスを予約、すなわちリザーブ状態としておくことは、“常駐終了”と呼ばれる。常駐終了はPUからの命令に応じて行われる。

【0091】

図25、26A及び26Bは、1グループのAPU及びそれらと関連するサンドボックスを含む、ストリーミングデータ（ストリーミングMPEGデータなど）を処理するための専用パイプライン構造の設定を例示する。図25に示すように、このパイプライン構造の構成要素にはPE2502とDRAM2518とが含まれる。PE2502にはPU2504とDMAC2506とが含まれ、複数のAPUには、APU2508と、APU2510と、APU2512とが含まれる。広帯域バス2516によりDMAC2506はDRAM2518と接続される。DRAM2518には複数のサンドボックス（サンドボックス2520、サンドボックス2522、サンドボックス2524、サンドボックス2526など）が含まれる。

【0092】

図26Aに専用パイプラインを設定するためのステップを例示する。ステップ2610で、PU2504はネットワーク・アプレットを処理するようにAPU2508を割り当てる。ネットワーク・アプレットはネットワーク104のネットワーク・プロトコルの処理用プログラムを有する。この場合、このプロトコルは、伝送制御プロトコル/インターネット用プロトコル(TCP/IP)である。このプロトコルに従うTCP/IPデータ・パケットは、ネットワーク104を介して伝送される。受信時に、APU2508はこ

10

20

30

40

50

これらのパケットを処理し、パケット内のデータを組み立て、ソフトウェア・セル102の中へ入れる。ステップ2612で、PU2504は、ネットワーク・アプレットの処理の完了時に常駐終了を実行するようにAPU2508に指示する。ステップ2614で、PU2504は、APU2510及び2512がMPEGアプレットの処理を行うように割り当てる。ステップ2615で、PU2504はMPEGアプレットの処理の完了時に常駐終了を実行するようにAPU2510と2512に指示する。ステップ2616で、PU2504は、APU2508とAPU2510によるアクセス用ソース・サンドボックスとしてサンドボックス2520を指定する。ステップ2618で、PU2504はAPU2510によるアクセス用宛先サンドボックスとしてサンドボックス2522を指定する。ステップ2620で、PU2504は、APU2508とAPU2510によるアクセス用ソース・サンドボックスとしてサンドボックス2524を指定する。ステップ2622で、PU2504は、APU2512によるアクセス用宛先サンドボックスとしてサンドボックス2526を指定する。ステップ2624で、APU2510とAPU2512とは、それぞれ、ソース・サンドボックス2520とソース・サンドボックス2524の範囲内のメモリ・ブロックへ同期読み取りコマンドを送り、これらのメモリ・ブロックをブロッキング状態に設定する。最後に、処理はステップ2628へ移り、そこで専用のパイプラインの設定が完了し、パイプライン専用のリソースが予約される。このようにして、APU2508、2510、2512及びそれらと関連するサンドボックス2520、2522、2524、及び2526は予約状態に入る。

【0093】

図26Bにこの専用パイプラインによるストリーミングMPEGデータの処理ステップを例示する。ステップ2630で、APU2508はネットワーク・アプレットを処理し、そのローカル・ストレージの中でTCP/IPデータパケットをネットワーク104から受信する。ステップ2632で、APU2508はこれらのTCP/IPデータパケットを処理し、これらのパケット内のデータを組み立て、ソフトウェア・セル102の中へ入れる。ステップ2634で、APU2508はソフトウェア・セルのヘッダ2320(図23)をチェックし、セルがMPEGデータを含むかどうかの判定を行う。セルがMPEGデータを含まない場合、ステップ2636で、APU2508は専用パイプラインに含まれない他のAPUによって他のデータを処理するために、DRAM2518内に指定される汎用サンドボックスへセルを伝送する。APU2508はこの伝送についてPU2504に通知する。

【0094】

一方、ソフトウェア・セルがMPEGデータを含む場合は、ステップ2638で、APU2508はそのセルの前のセルID2330(図23)をチェックし、そのセルが属するMPEGデータ・ストリームを識別する。ステップ2640で、APU2508はセル処理用の専用パイプラインのAPUを選択する。この場合、APU2508はこれらのデータを処理するAPU2510を選択する。この選択は前回のセルID2330とロード・バランシング・ファクタ(負荷平衡係数)とに基づく。例えば、そのソフトウェア・セルが属するMPEGデータ・ストリームの前回のソフトウェア・セルが処理用としてAPU2510へ送られたことが前回のセルID2330によって示されている場合、現在のソフトウェア・セルも通常の処理用としてAPU2510へ送られる。ステップ2642で、APU2508は、サンドボックス2520へMPEGデータを書き込む、同期書き込みコマンドを出す。このサンドボックスはあらかじめブロッキング状態に設定されているので、ステップ2644で、MPEGデータはサンドボックス2520からAPU2510のローカル・ストレージへ自動的に読み出される。ステップ2646で、APU2510はそのローカル・ストレージでMPEGデータを処理してビデオ・データを生成する。ステップ2648で、APU2510はサンドボックス2522へビデオ・データを書き込む。ステップ2650で、APU2510は同期読み出しコマンドをサンドボックス2520へ出し、このサンドボックスに追加MPEGデータ受信の準備をする。ステップ2652で、APU2510は常駐完了処理を行う。この処理により、このAPUは予約

状態に入り、この予約状態の間、APUはMPEGデータ・ストリームの中で追加MPEGデータの処理を行うべく待機する。

【0095】

他のタイプのデータ処理用として、1グループのAPU及びそれらと関連するサンドボックス間でその他の専用構造の設定が可能である。例えば、図27に示すように、APUの専用グループ(APU2702、2708、2714など)を設定し、3次元オブジェクトに対して幾何学変換を実行して2次元ディスプレイ・リストの生成を行うことが可能となる。これらの2次元ディスプレイ・リストを他のAPUによって更に処理(レンダリング)、画素データの生成を行うようにすることが可能である。この処理を実行するために、3次元オブジェクトと、これらのオブジェクト処理から結果として生じるディスプレイ・リストの格納用として、サンドボックスが、APU2702、2708、2714の専用となる。例えば、ソース・サンドボックス2704、2710、2716は、それぞれ、APU2702、2708、2714によって処理された3次元オブジェクトの格納専用となる。同様に、宛先サンドボックス2706、2712、2718はそれぞれ、APU2702、APU2708、APU2714によるこれらの3次元オブジェクトの処理から結果として生じるディスプレイ・リストの格納専用となる。

【0096】

調整用APU2720は、そのローカル・ストレージにおける、宛先サンドボックス2706、2712、2718からのディスプレイ・リストの受信専用である。APU2720はこれらのディスプレイ・リスト間で調整を行い、画素データのレンダリングのためにこれらのディスプレイ・リストを他のAPUへ送る。

【0097】

システム1010のプロセッサは絶対タイマーも使用する。この絶対タイマーはAPUとPEの他のエレメントへクロック信号を出力する。このクロック信号はこれらのエレメントを駆動するクロック信号に依存せず、かつ、このクロック信号より高速である。この絶対タイマーの利用が図28に例示されている。

【0098】

この図に示すように、絶対タイマーによってAPUによるタスク・パフォーマンスのためのタイム・バジェット(割り当て時間)が決定される。このタイム・バジェットによって、これらのタスクの完了時間が設定されるが、この時間はAPUによるタスク処理に必要な時間より長い時間になる。その結果、各タスクについて、タイム・バジェットの範囲内に、ビジネ時間とスタンバイ時間とが存在することになる。全てのアプリは、APUの実際の処理時間あるいは速度にかかわらず、このタイム・バジェットに基づいて処理を行うように書かれる。例えば、PEの特定のAPU用として、タイム・バジェット2804のビジネ時間2802中に特定のタスクを行うことができる。ビジネ時間2802がタイム・バジェット2804未満であるため、スタンバイ時間2806はタイム・バジェット中に生じる。このスタンバイ時間中、APUはAPUが消費するパワーが少なくなるスリープモードに入る。

【0099】

タイム・バジェット2804が満了するまで、他のAPU又はPEの他のエレメントがタスク処理の結果を予想することはない。したがって、APUの実際の処理速度にかかわらず、絶対タイマーによって決定されるタイム・バジェットを用いて、APUの処理結果が常時調整される。

【0100】

将来、APUによる処理速度は更に高速になる。しかし、絶対タイマーによって設定されるタイム・バジェットは同じままである。例えば、図28に示すように、将来のAPUは更に短時間でタスクを実行することになり、したがって、スタンバイ時間は更に長くなるであろう。したがって、ビジネ時間2808はビジネ時間2802よりも短くなり、スタンバイ時間2810はスタンバイ時間2806よりも長くなる。しかし、絶対タイマーによって設定された同じタイム・バジェットに基づいて処理を行うようにプログラムが書

10

20

30

40

50

かれていますので、ＡＰＵ間の処理結果の調整が維持される。その結果、更に高速のＡＰＵが、その処理の結果が予測される時点でコンフリクトを生じることなく、低速のＡＰＵ用として書かれたプログラムの処理を行うことが可能となる。

【０１０１】

動作速度の向上や動作速度が異なることに起因するＡＰＵの並列処理の調整問題に対しては、ＡＰＵ間での調整を決定する絶対タイマーに代えて、ＰＵまたは１以上の指定ＡＰＵにおいて、ＡＰＵが実行している特定の命令（マイクロコード）の分析をアプレットの処理時に行うようにすることもできる。“オペレーションなし”（“NOOP”）命令を命令の中へ挿入し、ＡＰＵのいくつかによってこの命令を実行してアプレットによって予測されるＡＰＵによる処理を１ステップずつ適切に行うことが可能となる。命令の中へこれらのNOOPを挿入することにより、全ての命令のＡＰＵによる実行を行うための正しいタイミングの維持が可能となる。

【０１０２】

上述のように、各処理エレメント（PE）には少なくとも１つの処理ユニット（PU）と、１つ以上の複数の付加処理デバイス（APU）が含まれ、PUの調整及び制御下で、１つ以上のアプリケーションによりデータの並列処理がAPUによって実行される。これに関連して、また、手短かに図１８を参照すると、図１８にはAPUと共用メモリとの間にデータをやりとりして送るための、例示的なデータレベル同期（DLS）機構の状態で示されていた。しかし、前述のように、このDLS機構は外部デバイスへの、及び／又は外部デバイスからのデータの処理にも好適に利用される。更に、以下の説明からわかるように、１／０デバイスを扱うためにDLSを利用することにより、その他の利点が与えられる。例えば、現在のところそれらの利点としては、デバイス実装の詳細を隠す能力、複数のデバイスを扱う柔軟性、１つ以上の外部ソースからのストリーミング・データの処理を行うAPUの能力、が挙げられる。図２９に留意すると、図２９には本発明の概念による例示的なDLSアーキテクチャが示されている。

【０１０３】

図２９において、構成２９００にはPE２９００と、DRAM（又は共用メモリ）２９１０と、インターフェースデバイス（インターフェース）２９１５と、外部デバイス２９２０とが含まれる。PE２９００は１つ以上の処理エレメントを示し、さらにPE２９００には、少なくとも１つのPU２９０５と、１つ以上のAPU（APU２９０１、２９０２、２９０３により示す）とが含まれる。PE２９００はバス２９０６を介して共用メモリと結合されており、またPE２９００は前述のDLS機構とメモリ保護機構を用いて、DRAM２９１０へのデータの書き込みと、DRAM２９１０からのデータの読み出しを行う。本発明の概念によると、現在のところ、このDLS機構は外部デバイスと用いられるように拡張されている。そのために、また図２９からわかるように、PE２９００はバス２９０７を介してインターフェース２９１５と結合されており、一方でDRAM２９１０はバス２９０８を介してインターフェース２９１５と結合されている。またインターフェース２９１５は、バス２９１６を介して、図中で外部デバイス２９２０として示される１つ以上の外部デバイスと結合されている。外部デバイス２９２０は、ハードディスク又は光ディスクなどのメモリ、１／０プロセッサ又は画像処理用プロセッサなどのプロセッサ、トランシーバ（ネットワークランシーバなど）等を含む複数のデバイスのうちの任意の１つとすることができるが、外部デバイスはこれらに限定されるものではない。バス２９１６は外部デバイス２９２０をインターフェース２９１５と結合するための、１つ以上の通信パスを示し、それらは例えば、バケットネットワーク接続部、交換式ネットワーク接続部、シリアルバス、並列バス、などである。以下に更に説明しているように、バス２９０８はインターフェース２９１５を介して、DRAM２９１０へのアクセスを直接行う。

【０１０４】

図３０を参照すると、まず本発明の概念が、外部メモリ３０２０に関連して外部デバイスとして例示されている。特に、各APUはバス２９０７を介して、インターフェース２９１５へ要求を送り、及び／又は、バス２９０７を介してインターフェース２９１５から

10

20

30

40

50

の応答を受け取る、1つ以上のチャネルと関連付けられている。これについては図30に例示されており、APU2901に対し、チャネル0（要求）（2930）とチャネル1（応答）（2935）により関連付けられている。説明のため、エレメント2930と2935の双方とも、1つ以上のFIFO（first-in-first out）バッファである。簡素化のため、他のAPUにおける、順次転送される要求及び応答の他のチャネルは、図30の破線矢印で示す。ここでは、エレメント2930と2935にはコマンドと状態情報だけが格納されるとする。要求チャネルと応答チャネルはバス2907を介して伝送コントローラ（transfer controller）2945を用いてコマンドと状態情報を交換する。説明のため、伝送コントローラ2945はAPUからの要求の処理を行う外部DMACである。図30からわかるように、インターフェース2915には保護テーブル2940が含まれる。これは、PU2905の内部DMAC（図示せず）において図19と関連して上述した、キー管理テーブル1902に追加するものである。外部保護テーブル（保護テーブル2940など）の利用により、外部メモリ3020から、DRAM2910の1つ以上のサンドボックスへの直接的データ・ストリーミングが改善される。説明のため、保護テーブル2940の値は、上述の方法と同様の方法でPUにより割り当てられる。保護テーブル2940を利用して、伝送コントローラ2945は、DRAM2910の1つ、又は複数のサンドボックスと外部メモリ3020との間で、いずれかの方向にデータ通信を行うために、伝送バス2950を制御する。外部保護テーブルは必要とされない点に留意されたい。しかし、外部保護テーブルがない場合、好適にはPUから、例えば、PUと関連付けられて同様に設けられた要求／応答チャネルから、適切な許可が送られるのが望ましい。上述のように、PUは「信頼できるプログラム(trusted program)」を実行するため、関連の読み出し／書き込み許可の割り当てがなされるべきである。更に、保護テーブル2940が拡張され、外部デバイスへのアクセス、又はその1部へのアクセスを制御することもできる。これについては外部メモリ3020の特定のメモリ・アドレス領域により示される。

【0105】

引き続き図30を参照し、また図31と32を特に参照すると、外部メモリ3020からAPU2901のローカル・メモリ（図示せず）へのデータ伝送に利用するDLS法が例示されている。この方法は図24について上述した方法の変形であるので、全てのステップを以下に繰り返さず、また図31には全てのステップを示していない。この例では、APU2901によってアプレット（前述の、ソフトウェアの種類）が実行されると仮定する。このAPUは外部メモリ3020に現在格納されているデータ上動作する必要があるものである。まず図31のステップ3105を見ると、PU2905はAPU2901によって用いられるDRAM2910のメモリの割り当てを行う。更にステップ3110で、PU2905はバス2907を介して、割り当てられたメモリに対する適切なキー保護値をインターフェース2915の伝送コントローラ2945へ与え、DRAM2910のメモリへのアクセス制御を行う保護テーブル2940で用いる。PUはAPU2901によってアクセス制御が行われるように、そのキー管理テーブルの更新を行うことに加え、PUは上記処理を、図19に関連して上述した方法で行う。ステップ3115で、PU2905はAPU2901によりアプレットの実行を開始する。このアプレットは現在のところ外部メモリ3020に格納されているデータへのアクセスを要求する。APU2901を参照すると、ステップ3130で、このAPUにより、チャネル0を介してインターフェース2915へ要求が送られ、外部メモリ3020などの外部デバイスから、DRAM2910へのデータ伝送が行われる。この要求は、DRAM2910への同期書き込みオペレーションに対するものであり、またこの要求は前述のDMAコマンドと形式が類似している。説明のため、この要求には、例えば外部デバイス（ここでは、外部メモリ3020）や、利用される外部デバイスの一部（ここでは、外部メモリ3020のデータ位置）の識別子、伝送量、DRAM2910の対応アドレス、及び、オペレーションの種類が、読み出しオペレーションか書き込みオペレーションか（ここでは、DRAM2910への同期書き込みオペレーション）、が挙げられる。デバイスや利用されるデバイスの

10

20

30

40

50

一部などの外部デバイス情報やサイズ情報は、例えばアプレット関連情報として与えられる。また、他の種類の外部デバイス・パラメータには、デバイス番号やポート番号、仮想チャネル識別子などが挙げられるが、これらに限定されるものでないことに留意されたい。

【0106】

図32、及び引き続き図30を参照すると、インターフェース2915の伝送コントローラ2915により、ステップ3205において、APU2901からの伝送要求が受け取られる。ステップ3210で、伝送コントローラ2945により保護テーブル2940に格納された値のチェックが行われ、DRAM2910で要求されたメモリの値域が有効であるかが判断される。メモリの値域が無効である場合、ステップ3220で、伝送コントローラ2945により、チャネル1を介した伝送が拒否される。しかし、メモリの値域が有効である場合、ステップ3225で、伝送コントローラにより伝送が開始され、完了するまで伝送バス2950を介して、DMAモードでデータ伝送が継続して行われる（ステップ3230）。伝送が完了すると、伝送コントローラ2945はステップ3235で、応答チャネル1を介して伝送完了メッセージを送る。外部デバイス情報も保護テーブル2940に格納される場合は、ステップ3210や3215などで、APU2901が伝送を進める前に、その外部デバイスが利用許可を有しているかどうかのチェックも行う。

【0107】

本発明によれば、外部メモリ3020からAPU2901へのデータ伝送は、図18に関連して上述しているように、DLSを用いて行われる。これに関連して、図33にはDLSの用途が例示されており、図33ではデータ・フロー（破線矢印により示す）が示されている。同期書き込みオペレーションはインターフェース2915を通じてDRAM2910へ行われ、一方で同期読み出しオペレーションはAPU2901を通じて行われ、ローカル・メモリ2981への格納用に、DRAM2910からデータの読み出しが行われる。図17Bから17Dに関連して上述しているように、DRAM2910による個々の同期書き込みオペレーションと同期読み出しオペレーション処理とが、共用メモリのエンティティ状態、ブロッキング状態、及び、フル状態によって制御される（例えば、関連する状態情報のLSアドレスフィールド、APU ID、及びF/Eビット値、により示す）。したがって、例えば、最初にPUがAPUによって用いられるDRAMへのデータ伝送を行うといったことを必要とせずに、外部デバイスからAPUのローカル・メモリへ、効率的に、つまり低オーバーヘッドでデータの伝送を行うことが可能である。換言すれば、PUで実行されるべきI/O処理が、安全かつ同期して、APUにより実行されるアーキテクチャレベルにまで、その実行レベルが引き下げられている。これにより、複数のストリーミングアプリケーションにおいて、APUが、外部メモリの複数のバッファからのデータを素早く読出し、動作することが可能となる。

【0108】

同期読み出しオペレーション、つまり、DRAM2910から外部デバイス（外部メモリ3020など）への伝送、も上述と同様の方法で可能であることに留意されたい。これについては、図34に例示されており、また引き続き図31、32のフローチャートを参照する。例えば、ステップ3115で、PU2905は、アプレットのAPU2901により実行を開始する。このアプレットは、外部メモリ3020に格納するためにデータの伝送を必要とする。そのため、同期読み出しオペレーションに対する要求が、チャネル0を介してAPUによってインターフェース2915へ送られる（つまり、DRAM2910から外部デバイス（外部メモリ3020）へのデータ伝送）。この要求は前述のDMAコマンドと同様の形式である。この要求には、例えば外部デバイス（ここでは、外部メモリ3020）や、利用される外部デバイスの一部（ここでは、外部メモリ3020のデータ位置）の識別子、伝送量、DRAM2910の対応アドレス、及び、外部デバイスオペレーションが、読み出しオペレーションか書き込みオペレーションか（ここでは、DRAM2910からの同期読み出しオペレーション）、が挙げられる。

【0109】

10

20

30

40

50

これに関連して、DLSの利用により、インターフェース2915が同期方式でDRAM2910からのデータの読み出すのを可能にし、一方で、APU2901はDRAM2910への書き込みを行う。これについては図34に示されており、APU2901のローカル・メモリ2981から外部デバイスへのデータ・フロー（破線矢印により示す）が図34に示されている。これに関連して、インターフェース2915がデータの読み出しを試みたかどうかに関わらず、DLSの様々な状態がF/Eビット値により示されている。このため、DRAM2915の関連メモリ・ロケーションに対する上述の状態情報が、図35に示すように変更される。この状態情報には、上述のようにF/Eビット3501と、デバイスID3502と、データが伝送される伝送先であるアドレス3503が含まれる。デバイスID3502により、APUなどのデバイスのタイプが識別され、又はインターフェースなどの別のデバイスが識別される。これに関連して、インターフェース2915にはすでに保護テーブルを持つDMAタイプのコントローラが含まれているので、アドレス・フィールドは要求され得ない点に留意すべきである。つまり、DRAM2910は、同期読み出しオペレーションが許可された場合に、インターフェース2915（デバイスID3502により識別される）へ単に通知を出すだけである。

【0110】

上述のように、図30により示されている構成は単に例示的なものであり、その他のタイプの外部デバイスへ直接的に拡張することができる。例えば、図36を検討する。この図は図30と同じものである。このため同じ参照符号は同様の要素を表しており、そのような要素についてはここではさらに説明しない。図36では、外部デバイスは物理レシーバ（physical receiver: PHY）3620によって示される。物理レシーバ3620はバス3621を介して、イーサネットなどのネットワーク（図示せず）からパケット通信を受信し、受信したビット・ストリームのデコーディングを行い、パケットのストリームを与える。この例では、APUによって実行されたアプレットは受信したパケットストリームのリンク層処理を示す。つまり、APU2901はPHY3620からのデータの処理を行うために上述のDLS機構を利用して、1部のプロトコルスタック（TCP/IP（通信制御プロトコル/インターネットプロトコル）など）に対し処理を行う。

【0111】

引き続き図36を参照し、また特に図37を参照すると、例示的なハンドシェーキング・シーケンスが示されており、外部デバイスを用いたデータレベル同期の利用がさらに例示されている。この図では、共有メモリ（DRAM2910など）の対応部分がエンプティ状態で開始されたと仮定する。最初に、APU（APU2901など）が要求チャネルを介して、外部デバイスから共有メモリの識別部分への同期書き込みオペレーションを要求するコマンドを、インターフェース（インターフェース2915など）へ送る。ここでは、バスインターフェース2915のチェックを行う保護テーブルが、外部デバイスを用いて伝送を開始すると仮定する。一方で、APUは共有メモリのこれらの部分からの同期読み出しオペレーションの実行を試みる。インターフェースによって書き込みが行われているデータはまだないので、共有メモリの対応部分がブロッキング状態に入る。この状態で、前述のように、APUに対するローカル・メモリのアドレスが共有メモリのこれらの部分に対し、関連する状態情報フィールドに格納される。続いて、インターフェースが共有メモリの識別部分への同期書き込みオペレーションを行った後、APUに対し、同期読み出しオペレーションが、格納されたアドレス値により示されるアドレスへ行われる。その結果、共有メモリのこれらの部分がエンプティ状態へ戻る。その他の順列に対する状態変化（図18に示す）を示す、同様のハンドシェーキング・シーケンスは簡明であり、ここには説明していない。

【0112】

1つ以上のAPUからの要求は、インターフェースを通じて、各要求の完了を待つ必要がないことがわかる。例えば、APUは1つ以上の外部デバイスからのデータに対し、インターフェースへ複数の要求を出すことができる。DLSの利用により、（インターフェースを介して）APU、あるいは外部デバイスのいずれから、同期読み出しオペレーション

ンが行われるまで、データが確実に共用メモリに保護される。そのため、DLSの利用により、APUと1つ以上の外部デバイスとの間に実行されるバースト伝送と並列バースト伝送とが可能になる。

【0113】

説明のため、図36に示すデバイスを用いた本発明の概念の他の適用例を図38に示す。APU2901はインターフェース2915を介して、多数の外部デバイスと接続している。これらの外部デバイスは、USB（ユニバーサル・シリアル・バス）デバイス、IEEE1394（一般に“ファイアワイヤ”と呼ばれる）デバイス、シリアル・デバイス、ハードディスク、ネットワーク・デバイス（ネットワークランシバなど）、光ディスク、である。その他の外部デバイス（図示せず）としては、I/Oプロセッサ、又は画像処理用プロセッサなどのプロセッサが挙げられる。APU2901は1つ以上のこれらの外部デバイスからのデータの読み出しを行うために、要求4301を送る。そのため、各デバイスはインターフェース2915の制御を介して、続いてDRAM2910のそれぞれに割り当てられたメモリ・ロケーションへ同期書き込みオペレーションを行う。DRAM2910のメモリ・ロケーションは、メモリ・ロケーション4311、4312、4313、4314、4315、4316により示される（各メモリ・ロケーションには、関連付けられた状態情報が含まれることも示し、それらは単にF/Eビットにより示される）。APU2901は、同期読み出しオペレーションを行うことにより、バス2906を介して、DRAM2910からデータを読出す。

【0114】

図39を参照すると、本発明の概念の別の実施形態が例示されている。図39に簡略化したストリーミング構成3700を示す。構成3700には、PE3790と、DRAM（又は共用メモリ）3710と、インターフェースデバイス（インターフェース）3715と、物理トランシバ（PHY）により示される外部デバイスとが含まれる。外部デバイス3720はインターフェース3715の一部ともでき、例えば、インターフェース3715と結合することもできる。PE3790は1つ以上の処理エレメントを示し、更にPE3790には少なくとも1つのPU3705と、1つ以上のAPU（APU3701、3702、3703により示す）とが含まれる。PE3790はバス3706を介して共用メモリと結合されている。PE3790はまた、前述のDLS機構とメモリ保護機構を用いて、DRAM3710へのデータの書き込みと、DRAM3710からのデータの読み出しを行う。本発明の概念によれば、現在のところ、DLS機構は外部デバイスと用いられるように拡張されている。そのために、また、図39からわかるように、各APUは、バス3707を介してインターフェース3715と結合されている。特に各APUは、1つ以上のチャネルと関連付けられており、インターフェース3715への要求の送信、及び/又は、インターフェース3715からの応答の受信、及び/又は、インターフェース3715とのデータ通信が、バス3707を介して行われる。これについては図39に例示されており、APU3701に対して、チャネル0（入力）（3730）と、チャネル1（出力）（3735）が関連付けられている。説明のため、エレメント3730と3735の双方が、1つ、またはそれ以上のFIFOバッファである。その他のAPUに関しては、バス3707を介して転送されるそれらの入力及び出力のチャネルは、図39に破線矢印により示される。

【0115】

インターフェース3715には、チャネル・インターフェース・エレメント3745と3750とが含まれ、インターフェース3715により、APU3701とPHY3720との間のそれぞれの入出力データストリームが結合される。説明のため、チャネル・インターフェース・エレメント3750はトランシバであり、(a) APU3701の出力チャネルから送られるデータを、バス3707と3716を介してエンコードし、バス3721を介して伝送する、及び(b) バス3721から受け取ったデータをデコードし、バス3716と3707を介して、APU3701の入力チャネルへデコードしたデータを送る。バス3721は外部デバイス3720を、パケットネットワーク接続部、交換

式ネットワーク接続部、シリアルバス、並列バス、などへ結合するための、1つ以上の通信バスを示す。

【0116】

上述のように、インターフェース3715には、チャンネル・インターフェース・エレメント3745と3750が含まれる。チャンネル・インターフェース・エレメント3745はAPU3701から出力チャンネルを受信する。本発明によれば、出力チャンネル自体には制御チャンネルとデータチャンネルとが含まれる。この制御チャンネルにより、上述の要求と応答のチャンネルが形成され、一方で、データのチャンネルにより、外部デバイスへ送られるデータが転送される。この例では、要求チャンネルにより外部デバイス識別子（ここではPHY3720）を含むコマンド情報や、実行されるオペレーション（ここでは、書き込みオペレーション）などが転送される。出力チャンネルのデータチャンネル部は、PHY3720によってエンコードされ、バス3721を介して伝送されるデータを示す。同様に、チャンネル・インターフェース・エレメント3745は、APU3701の入力チャンネルへ結合される。入力チャンネル自体には、制御チャンネルとデータチャンネルとが含まれる。制御チャンネルにより、上述の要求と応答のチャンネルが形成され、一方で、APUへ送られるデータはデータチャンネルによって転送される。この例では、外部デバイス識別子（ここでは、PHY3720）や、実行されるオペレーション（ここでは、読み出しオペレーション）などを含むコマンド情報が、要求チャンネルによって転送される。この入力チャンネルのデータチャンネル部により、バス3721から受信した信号受信の結果として、PHY3720によってデコードされたデータが示される。

【0117】

この実施形態では、インターフェース3715内には保護テーブルが必要とされないことがわかる。特に、また更に以下に説明しているように、図18に例示している前述のDLS機構とメモリ保護機構により、各APUはDRAM3710からのデータの読み出しと、DRAM3710へのデータの書き込みとが行われる。

【0118】

引き続き図39を参照すると、外部デバイスからAPU2901のローカル・メモリ（図示せず）へのデータ伝送に用いる例示的な方法が図40に示されている。この方法により、上述した図24の方法も実行されると仮定する。その方法とは、PUによる保護テーブル値の割り当て、APU3701などへのアプレットの割り当て、などである。説明のため、この例では、APU3701によって実行されるアプレットは、受信したパケットストリームのリンク層処理と、送信したパケットストリームに対するリンク層処理とを示す。つまり、APU2901によって上述のDLS機構を利用したプロトコルスタック（TCP/IPなど）の一部分に対する処理が行われる。

【0119】

図40のステップ3830をまず参照すると、APU3701は、バス3707を介して、PHY3720からの、また、PHY3720へのデータ伝送要求を、インターフェース3715へ行う。この例では、入力チャンネルと出力チャンネルの双方の命令部を介してこの要求が行われる。しかし、入力チャンネル及び出力チャンネルの双方の利用は必要とされない。この要求には、外部デバイス（ここではPHY3720）の識別子などが含まれ、また、利用する外部デバイスの一部（特定のポートなど）や、外部デバイスのオペレーションの種類（ここでは、入力チャンネルに対する外部デバイスからの読み出しと、出力チャンネルに対する外部デバイスへの書き込み）も含まれる。デバイスや利用するデバイスの一部などの外部デバイス情報は、アプレット関連の情報などに与えられるとする。さらに、他の種類の外部デバイスパラメータには、デバイス番号やポート番号、仮想チャンネル識別子、などを挙げられるが、これらに限定されるものではない。

【0120】

ステップ3805で、インターフェース3715は要求を受け取り、PHY3720への、またPHY3720からの伝送を開始する。特定のハンドシェーキングがPHY3720に必要とされる範囲で、このハンドシェーキングがインターフェース3715によつ

て行われ、また、これがAPU3701にとってトランスバレントなものであるとする。ステップ3710で、インターフェース3715は伝送を開始し、伝送が完了するまで、PHY3720からのデータ伝送を継続する(ステップ3815)。このようなデータ伝送は、例えば遠方終端への接続が切れるまでは継続される。この、PHY3720への、またPHY3720からのデータ伝送の部分は、外部デバイスの種類に応じて、同期して、又は非同期に行うことができる。データは入力チャネルのデータチャネル部内、例えば、エレメント3735のFIFOの一部などへ、伝送される。伝送が完了すると、インターフェース3715はステップ3820で、入出力チャネルのコマンド部の応答チャネルを介して、伝送完了メッセージを送る。

【0121】

本発明によれば、APU3701による、外部デバイス3720からの、また、外部デバイス3720へのデータ伝送は、DLSを用いて行われる。このDLSの状態図は先述の図18に説明している。これに関連して、DLSの用途を図41に例示する。APUと外部デバイスとの間のデータ・フローが破線矢印により示されている。図41からわかるように、APU3701はバス3707による入力チャネルを介して、上述しているようにローカル・メモリの中にデータを直接受け取る。同様に、APU3701はバス3707による出力チャネルを介して、ローカル・メモリから直接データを送る。これにより、速くて低いオーバーヘッドの通信チャネルがもたらされる。例えば、PHY3720を介して送信されるデータは、まずDRAM3710に格納され得る。APU3701は同期読み出しオペレーションを介して、DLSを用いて、このデータをDRAM3710から抽出する。これについては図41に対応の破線矢印により例示している。次いで、APU3701はリンク層のアプレットによりこのデータの処理を行い、処理したデータ(フォーマットされたパケットのストリームなど)を(インターフェース3715を介して)PHY3720へ送り、更にエンコードを行い、伝送する。同様に、PHY3720によって受け取られたデータのデコードが行われ、データはローカル・メモリ3781に格納するために入力チャネルを介してAPU3701へ送られ、APU3701によって処理される。APU3701はリンク層のアプレット(ヘッダの除去、エラーチェックなど)によりデータの処理を行い、同期書き込みオペレーションを介して、DLSを用いて、処理されたデータをDRAM3710へ送る。これについては図41に、対応の破線矢印で描かれている。

【0122】

図42に図39に関連して示され、上記に説明されている、簡略化したストリーミング構成の変形を例示する。この実施形態は図39に示す方法と同様の方法で動作するため、全ての図については詳細に説明していない。図42の構成4000にはPE4090と、DRAM(又は、共用メモリ)4010と、インターフェース4015と、物理トランシーバ(PHY)4020と物理トランシーバ(PHY)4080により示される外部デバイス、とが含まれる。図42からわかるように、説明のため、インターフェース4015は外部デバイス4020及び4080と結合されている。PE4090は1つ以上の処理エレメントを示し、更に、PE4090には少なくとも1つのPU4005と、1つ以上のAPU(APU4001、4002、4003)とが含まれる。PE4090はバス4006を介して共用メモリと結合されており、前述のDLSとメモリ保護機構を用いて、DRAM4010へのデータの書き込みと、DRAM4010からのデータの読み出しとを行う。図42からわかるように、APUはバス4007を介して、インターフェース4015と結合されている。特に、各APUは1つ以上のチャネルと関連付けられており、バス4007を介して、インターフェース4015への要求の送信、及び/又は、インターフェース4015からの応答の受信、及び/又はインターフェース4015とのデータ通信を行う。これについては図42に例示されており、APU4001に対し、入力/出力チャネル4050と入力/出力チャネル4055が関連付けられている。説明のため、これらのエレメントは双方とも、1つ以上のFIFOバッファである。バス4007を介して形成されるその他のAPU用のその他の入力及び出力のチャネルは、図42の破線矢

印により示されている。

【0123】

インターフェース4015にはエレメント4030と4035とが含まれ、各エレメントにはそれぞれ、2つのチャネル・インターフェース・エレメント4031、4032と4036、4037とが含まれる。(これらは図39で説明している種類である。)これらのエレメントにより、データストリームがそれぞれの外部デバイス(ここでは、4020と4080)へ結合される。説明のため、この外部デバイスは、それぞれのバス4021と4081への、及び、それぞれのバス4021と4081からのデータ通信用トランシーバである。この外部デバイス4020、及び/又は4080には、例えばハードディスク又は光ディスクなどのメモリ、I/Oプロセッサ又は画像処理用プロセッサなどのプロセッサ、トランシーバ(ネットワークトランシーバなど)なども挙げられる。

10

【0124】

上述の通り、この例では、APU4001は2つの入力/出力チャネル(4050と4055)へのアクセスを持ち、各チャネルには更に、コマンド情報とデータを転送する、上述の入出力チャネルが含まれる。上述した方法と同様の方法で、APU4001によってアプレットが実行され、外部デバイス(4020と4080)との間にデータが転送される。APU4001は複数の入力/出力チャネルへのアクセスを持つので、APU4001がDRAM4010へアクセスすることは要求されない。そのために、図42に示す変形は、外部デバイス間にデータを転送する、効率的で高速な構成を提供する。

【0125】

このように、上記の内容は、単に本発明の原則を例示するものに過ぎず、当業者は、ここには明示的に記載されていないが、本発明の趣旨及び範囲から逸脱することなくその原則を具体化する数多くの代替の構成を考案しうるのであることを理解されよう。例えば、本発明の概念が、データレベルの同期化を行う入力/出力インターフェースを探り上げて説明されているが、インターフェースが必ずしも入力と出力の双方を行う必要はなく、例えば、インターフェースは入力デバイスのみ、又は、出力デバイスのみ、又は、一般の入力/出力ネットワークで用いられる、入力デバイスと出力デバイスの任意の組合せ、などのうちの1つでありうる。

20

【図面の簡単な説明】

【0126】

- 【図1】本発明によるコンピュータ・ネットワークのアーキテクチャ全体を示す説明図。
- 【図2】本発明によるプロセッサ・エレメント(PE)の構造を示す説明図。
- 【図3】本発明による広帯域エンジン(BE)の構造を示す説明図。
- 【図4】本発明による付加処理デバイス(APU)の構造を示す説明図。
- 【図5】本発明によるプロセッサ・エレメントと、ビジュアルライザ(VS)と、光インターフェースとの構造を示す説明図。
- 【図6】本発明によるプロセッサ・エレメントの1つの組合せを示す説明図。
- 【図7】本発明によるプロセッサ・エレメントの別の組合せを示す説明図。
- 【図8】本発明によるプロセッサ・エレメントの更に別の組合せを示す説明図。
- 【図9】本発明によるプロセッサ・エレメントの更に別の組合せを示す説明図。
- 【図10】本発明によるプロセッサ・エレメントの更に別の組合せを示す説明図。
- 【図11A】本発明によるチップ・パッケージ内へ光インターフェースを統合した例を示す説明図。

30

【図11B】図11Aの光インターフェースを用いるプロセッサの1つの構成を示す説明図。

【図11C】図11Aの光インターフェースを用いるプロセッサの別の構成を示す説明図。

40

【図12A】本発明によるメモリ・システムの構造を示す説明図。

【図12B】本発明による第1の広帯域エンジンから第2の広帯域エンジンへのデータの書き込みを示す説明図。

50

- 【図 13】本発明によるプロセッサ・エレメント用の共用メモリの構造を示す説明図。
【図 14 A】図 13 に示すメモリ・バンクの一構造を示す説明図。
【図 14 B】図 13 に示すメモリ・バンクの別の構造を示す説明図。
【図 15】本発明による DMA C の構造を示す説明図。
【図 16】本発明による DMA C の代替の構造を示す説明図。
【図 17 A】本発明によるデータ同期オペレーションを示す説明図。
【図 17 B】本発明によるデータ同期オペレーションを示す説明図。
【図 17 C】本発明によるデータ同期オペレーションを示す説明図。
【図 17 D】本発明によるデータ同期オペレーションを示す説明図。
【図 17 E】本発明によるデータ同期オペレーションを示す説明図。
【図 17 F】本発明によるデータ同期オペレーションを示す説明図。
【図 17 G】本発明によるデータ同期オペレーションを示す説明図。
【図 17 H】本発明によるデータ同期オペレーションを示す説明図。
【図 17 I】本発明によるデータ同期オペレーションを示す説明図。
【図 17 J】本発明によるデータ同期オペレーションを示す説明図。
【図 17 K】本発明によるデータ同期オペレーションを示す説明図。
【図 17 L】本発明によるデータ同期オペレーションを示す説明図。
【図 17 M】本発明によるデータ同期オペレーションを示す説明図。
【図 17 N】本発明によるデータ同期オペレーションを示す説明図。
【図 17 O】本発明によるデータ同期オペレーションを示す説明図。
【図 18】本発明によるデータ同期方式によるメモリ・ロケーションの様々な状態を示す説明図。
【図 19】本発明によるハードウェア・サンドボックス用のキー管理テーブルの構造を示す説明図。
【図 20】本発明によるハードウェア・サンドボックス用のメモリ・アクセス・キーの格納方式を示す説明図。
【図 21】本発明によるハードウェア・サンドボックス用のメモリ・アクセス管理テーブルの構造を示す説明図。
【図 22】図 19 のキー管理テーブルと、図 21 のメモリ・アクセス制御テーブルとを用いてメモリ・サンドボックスにアクセスするステップを示すフローチャート。
【図 23】本発明によるソフトウェア・セルの構造を示す説明図。
【図 24】本発明による、A P U へ遅隔処理命令を出すステップを示すフローチャート。
【図 25】本発明によるストリーミング・データ処理専用パイプラインの構造を示す説明図。
【図 26 A】本発明によるストリーミング・データの処理時の図 25 の専用パイプラインによって実行されるステップを示すフローチャート。
【図 26 B】本発明によるストリーミング・データの処理時の図 25 の専用パイプラインによって実行されるステップを示すフローチャート。
【図 27】本発明によるストリーミング・データ処理専用パイプラインの他の構造を示す説明図。
【図 28】本発明による A P U によるアプリケーションとデータの並列処理を調整するための絶対タイマー方式を示す説明図。
【図 29】本発明の原則に従って、データレベル同期を用いた構成の実施形態を例示的に示す説明図。
【図 30】本発明の原則に従って、データレベル同期を用いた構成の実施形態を例示的に示す説明図。
【図 31】本発明の原則に従って、データレベル同期実行時に用いるフローチャートを例示的に示す説明図。
【図 32】本発明の原則に従って、データレベル同期実行時に用いるフローチャートを例示的に示す説明図。

10

20

30

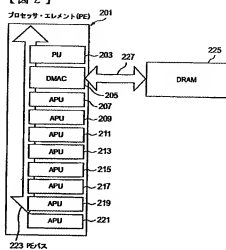
40

50

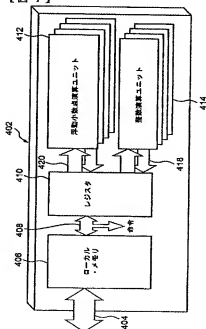
- 【図 3 3】 A P U と外部デバイスとの間にデータレベル同期の利用を示す説明図。
- 【図 3 4】 A P U と外部デバイスとの間にデータレベル同期の利用を示す説明図。
- 【図 3 5】 本発明の原則に従ってデータレベル同期状態情報を例示的に示す説明図。
- 【図 3 6】 本発明の原則に従ってデータレベル同期を用いた構成のもう 1 つの実記形態を例示的に示す説明図。
- 【図 3 7】 ハンドシェーキングシーケンスを例示的に示す説明図。
- 【図 3 8】 本発明の原則に従って、データレベル同期を用いた構成用の別の実施形態を例示的に示す説明図。
- 【図 3 9】 本発明の原則に従い、データレベル同期実行時に用いるフローチャートを例示的に示す図。
- 【図 4 0】 本発明の原則に従い、データレベル同期の実行に用いる別のフローチャートを例示的に示す説明図。
- 【図 4 1】 A P U と外部デバイスとの間に別のデータレベル同期の用途を例示的に示す説明図。
- 【図 4 2】 本発明の原則に従いデータレベル同期を用いた別の構成用の実施形態を例示的に示す説明図。
- 【符号の説明】
- 【0127】
- 101 システム
- 1010 キー
- 102 セル
- 104 ネットワーク
- 106 クライアント
- 108 サーバー・コンピュータ
- 1104 光インターフェース
- 1108 バス
- 1118, 1122 ポート
- 1126 光導波路
- 1160, 1162, 1164, 1166, 1182, 1184, 1186, 1188,
- 1190 光インターフェース
- 1206, 1234, 1242 コントロール
- 1212, 1240 ユニット
- 1221 クロスバ交換機
- 1232 外部ポート
- 1244, 1414, 1416 バンク
- 1406 ブロック
- 1504 ノード
- 1607, 1608 バス
- 1722 制御回路
- 1724, 1742 制御論理回路
- 1726 ストレージ
- 1728, 1731, 1732, 1746, 1750 ロケーション
- 1729, 1752, 1760, 1762 セグメント
- 1880 エンpty状態
- 1882 フル状態
- 1884 ブロッキング状態
- 1902 キー管理テーブル
- 1906 キー
- 1908 マスク
- 2006 格納位置

2008, 2010	セグメント	
2012	キー	
2102	アクセス管理テーブル	
2106	アドレス	
2110	キー	
2110	キー・マスク	
223	バス	
227	高帯域メモリ接続部	
2302, 2320	セル	
2308	ヘッダ	10
2322	インターフェース	
2332	実行セクション	
2334	リスト	
2520, 2522, 2524, 2526, 2704	サンドボックス	
2706	宛先サンドボックス	
2900	処理環境	
2901, 2902, 2903	A P U	
2905	P U	
2906~2908	バス	
2910	D R A M	20
2915	インターフェース	
2920	外部デバイス	
2930	エレメント	
2940	保護テーブル	
2945	伝送コントローラ	
2950	伝送バス	
2981	メモリ	
3020	外部メモリ	
3620	物理レシーバ	
3706, 3707	バス	30
3715	インターフェース	
3720	外部デバイス	
3730, 3735, 3745, 3740	エレメント	
3781	メモリ	
4015	インターフェース	
4020	外部デバイス	
4030, 4031	エレメント	
4050, 4055	出力チャネル	

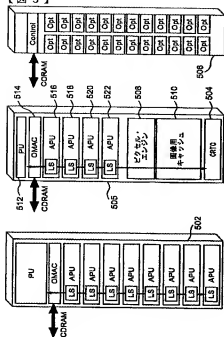
【圖 2】



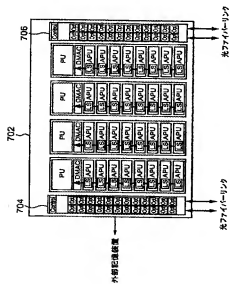
【圖 4】



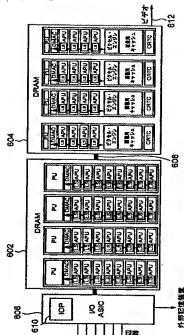
【図 5】



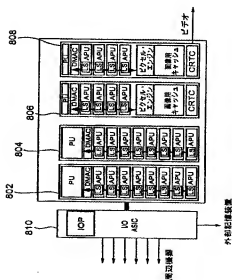
【図 7】



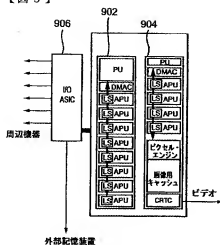
【図 6】



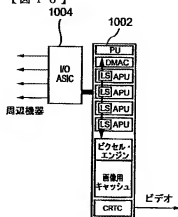
【図 8】



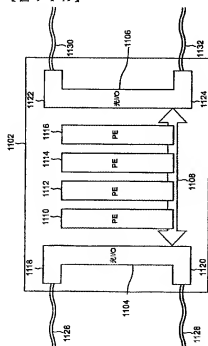
【図 9】



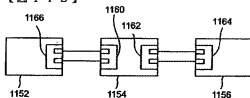
【図 10】



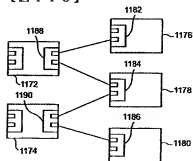
【図 11 A】



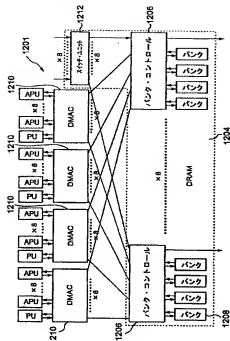
【図 11 B】



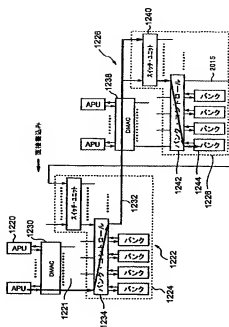
【図 11 C】



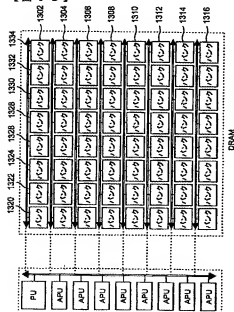
【図12A】



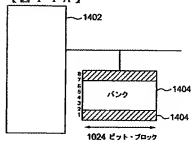
【図12B】



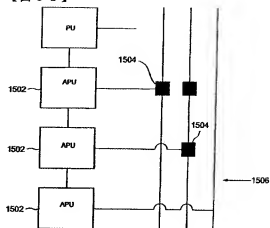
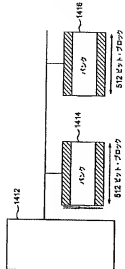
【図13】



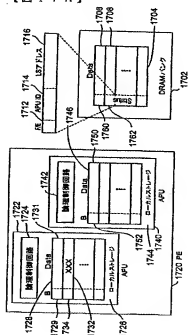
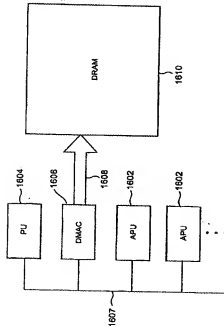
【図14A】



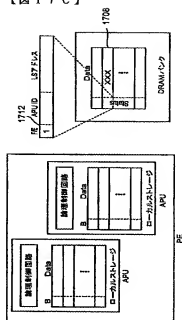
【圖 15】



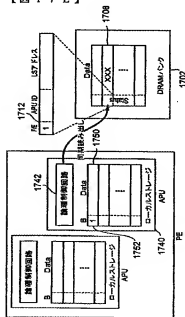
【圖 17 A】



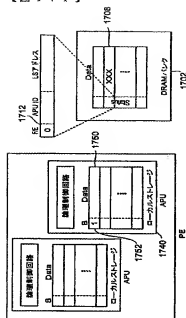
【☒ 1 7 C】



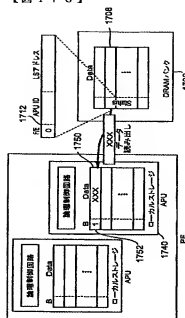
【圖 17 E】



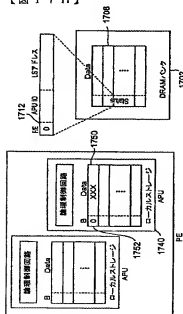
【図17F】



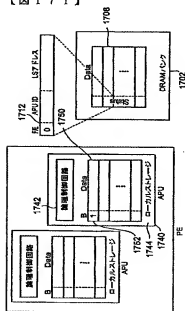
【図17G】



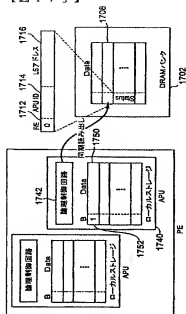
【図17H】



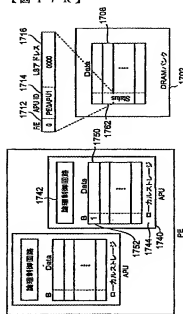
【図17I】



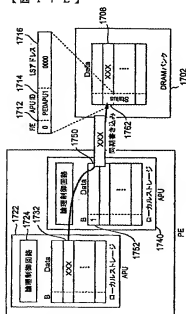
【図 17 J】



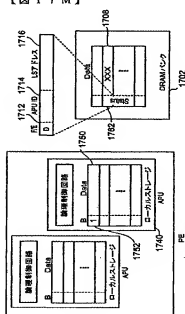
【図 17 K】



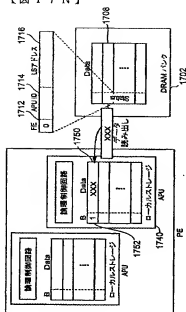
【図 17 L】



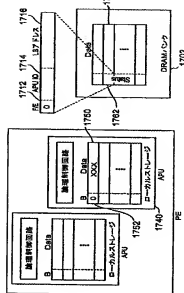
【図 17 M】



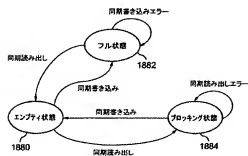
【図17N】



【図17O】



【図18】

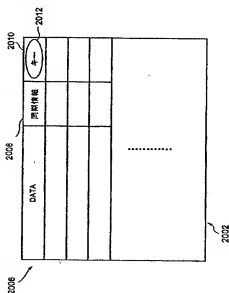


【図19】

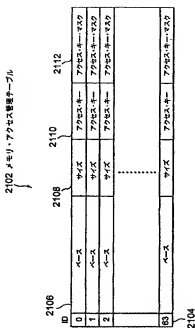
キー管理テーブル 1902

ID	1906	1908
0	APUキー	キー・マスク
1	APUキー	キー・マスク
2	APUキー	キー・マスク
...
7	APUキー	キー・マスク

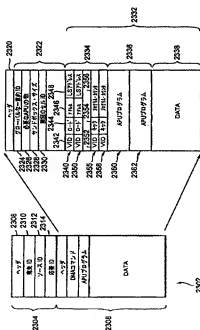
【図20】



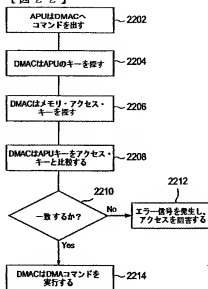
【圖 2 1】



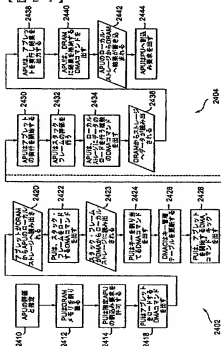
【图 2 3】



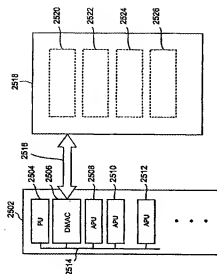
【圖 22】



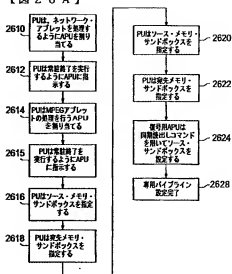
【图 24】



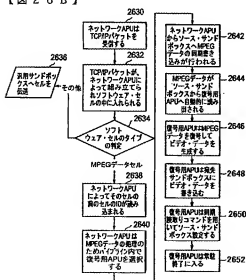
【図25】



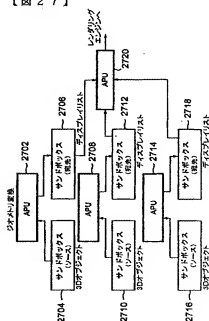
【図26A】



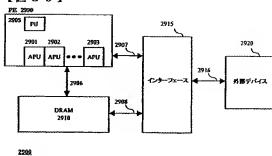
【図26B】



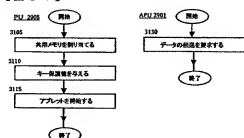
【図27】



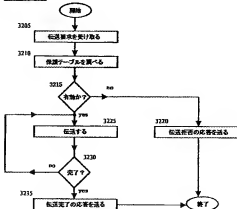
【☒ 29】



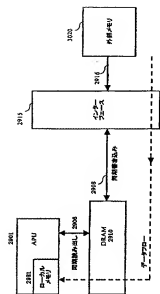
【 3 1 】



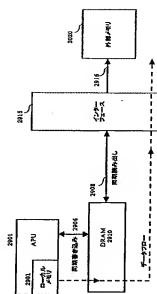
Interface 2012



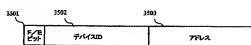
【图 3 3】



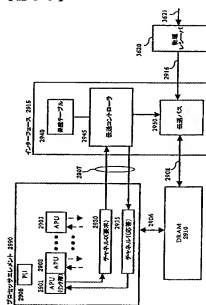
【图 3-4】



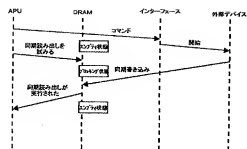
【圖 3 5】



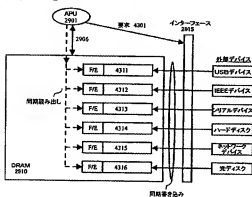
【图 3 6】



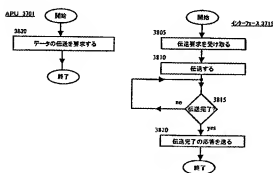
【図 37】



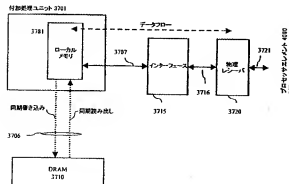
【図 38】



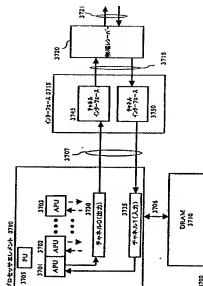
【図 40】



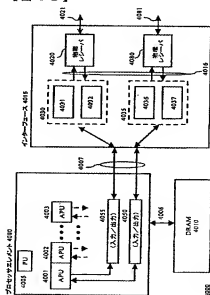
【図 41】



【図 39】



【図 42】



フロントページの続き

(72)発明者 山崎 剛

東京都港区南青山二丁目6番21号 株式会社ソニー・コンピュータエンタテインメント内